cadence™

Encounter® Text Command Reference

Product Version 8.1.2
June 2009

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

7
Floorplan Commands

14
Low Power Commands

About This Manual

The Cadence® Encounter® family of products provides an integrated solution for an RTL-to-GDSII design flow. This manual describes the syntax for the Encounter text commands.

The chapters are presented alphabetically by general command functionality. Within each chapter, the text commands appear alphabetically.

Audience

This manual is written for experienced designers of digital integrated circuits. Such designers must be familiar with design planning, placement and routing, block implementation, chip assembly, and design verification. Designers must also have a solid understanding of UNIX and Tcl/Tk programming.

Conventions Used in This Manual

This section describes the typographic and syntax conventions used in this manual.

`text`	Indicates text that you must type exactly as shown. For example: `analyze_connectivity -analyze all`
`text`	Indicates information for which you must substitute a name or value. In the following example, you must substitute the name of a specific file for *`configfile`*: `wroute filename` *`configfile`*

text	Indicates the following: ■ Text found in the graphical user interface (GUI), including form names, button labels, and field names ■ Terms that are new to the manual, are the subject of discussion, or need special emphasis ■ Titles of manuals
`[ ]`	Indicates optional arguments. In the following example, you can specify none, one, or both of the bracketed arguments: `command [-arg1] [arg2 value]`
`[ | ]`	Indicates an optional choice from a mutually exclusive list. In the following example, you can specify any of the arguments or none of the arguments, but you cannot specify more than one: `command [arg1 | arg2 | arg3 | arg4]`
`{ | }`	Indicates a required choice from a mutually exclusive list. In the following example, you must specify one, and only one, of the arguments: `command {arg1 | arg2 | arg3}`
`{[ ] [ ]}`	Indicates a required choice of one or more items in a list. In the following example, you must choose one argument from the list, but you can choose more than one: `command {[arg1] [arg2] [arg3]}`
`{ }`	Indicates curly braces that must be entered with the command syntax. In the following example, you must type the curly braces: `command arg1 {x y}`
`...`	Indicates that you can repeat the previous argument.
. . .	Indicates an omission in an example of computer output or input.

Command – Subcommand

Indicates a command sequence, which shows the order in which you choose commands and subcommands from the GUI menu.

In the following example, you choose *Floorplan* from the menu, then *Power Planning* from the submenu, and then *Add Rings* from the displayed list:

Floorplan – Power Planning – Add Rings

This sequence opens the Add Rings form.

Related Documents

For more information about the Encounter family of products, see the following documents. You can access these and other Cadence documents with the Cadence Help documentation system.

Encounter Foundation Flows Documentation

- *Encounter Foundation Flows User Guide*

  Describes how to use the scripts that represent the recommended implementation flows for digital timing closure with the Encounter software.

- *Encounter Foundation Flows: Flat Implementation Flow Guide*

  Describes the default-effort flat implementation flow, using the Encounter software.

- *Encounter Foundation Flows: Hierarchical Implementation Flow Guide*

  Describes the default-effort hierarchical implementation flow, using the Encounter software.

- *Encounter CPF-Based Low-Power Implementation Flow Guide*

  Describes the CPF-Based Low Power implementation flow, using the Encounter software.

Encounter Product Documentation

- *What's New in Encounter*

  Provides information about new and changed features in this release of the Encounter family of products.

- *Encounter Known Problems and Solutions*

  Describes important Cadence Change Requests (CCRs) for the Encounter family of products, including solutions for working around known problems.

- *Encounter User Guide*

  Describes how to install and configure the Encounter software, and provides strategies for implementing digital integrated circuits.

- *Encounter Menu Reference*

Provides information specific to the forms and commands available from the Encounter graphical user interface.

- *Encounter Database Access Command Reference*

  Lists all of the Encounter database access commands and provides a brief description of syntax and usage.

- *Encounter Library Development Guide*

  Describes library development guidelines for the independent tools that make up the Encounter family of products.

- `README` file

  Contains installation, compatibility, and other prerequisite information, including a list of Cadence Change Requests (CCRs) that were resolved in this release. You can read this file online at downloads.cadence.com.

6/3/09

1

Bus Plan Commands

createBusGuide

```
createBusGuide
     [-help]
     -netGroup netName
     -centerLine x1 y1 x2 y2
     -width value
     -layer {id | id1:id2}
     [-beginExt value]
     [-endExt value]
```

Specifies a bus guide segment.

Use this command after defining a net group associated with the new bus guide segment.

Parameters

`-beginExt` *value*

Specifies the width value, in microns, by which the bus guide segment is to be extended in the beginning.

`-centerLine` *x1 y1 x2 y2*

Specifies the coordinates of the center line of the bus guide segment.

The direction of the guide segment is derived from the `x1 y1 x2 y2` coordinate values.

Note: Non-orthogonal bus guide segments are not supported.

`-endExt` *value*

Specifies the width value, in microns, by which the bus guide segment is to be extended in the end.

`-help`

Outputs a brief description that includes type and default information for each `createBusGuide` parameter.

For a detailed description of the command and all of its parameters, use the man command:
`man createBusGuide`.

`-layer {`*id* | *id1*:*id2*`}`

Specifies the routing layer constraint honored by the router.

Routing done on non-preferred routing layers or on layers which are not part of the specified layer constraint can have high cost implications.

`-netGroup` *`netname`*

Specifies the name of the net group associated with the new bus guide segment that you create.

`-width` *`value`*

Specifies width value in microns.

Example

The following command creates a horizontal bus guide segment whose width is `120` microns, and layer constraint is from metal `4` to metal `8`:

```
createBusGuide -netGroup netGroup1 -centerLine 4421.8 10749.36 4960.8 10749.36 -
width 120 -layer 4:8
```

Related Topic

- "Bus Planning" chapter in the *Encounter User Guide*
  - Creating a Bus Guide

deleteBusGuide

```
deleteBusGuide
     [-help]
     {-all | -netGroup {netGroup | {list_of_net_groups}}}
```

Deletes a bus guide.

Parameters

`-all`	Deletes all the bus guides in the design.
`-help`	Outputs a brief description that includes type and default information for each `deleteBusGuide` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man deleteBusGuide`.
`-netGroup {netGroup | {list_of_net_groups}}`	Specifies the name of the net group or list of net groups associated with the bus guide segment(s) to be deleted.

deselectBusGuide

```
deselectBusGuide
     [-help]
     [-area x1 y1 x2 y2]
     [-netGroup {netGroup | {list_of_net_groups}}]
     [-layer {id | id1:id2}]
     [-direction {H | V}]
     [-all]
```

Deselects a bus guide segment.

If you do not specify any option, by default, the command deselects all the bus guide segments.

Use this command after selecting a bus guide segment.

Parameters

`-all`

Deselects all the bus guide segment in the design.

`-area` *x1 y1 x2 y2*

Specifies the coordinates of the area from which the bus guide segments are deselected.

`-direction {H | V}`

Specifies the direction of the bus guide segment to deselect. This can be `H` (horizontal) or `V` (vertical).

`-help`

Outputs a brief description that includes type and default information for each `deselectBusGuide` parameter.

For a detailed description of the command and all of its parameters, use the man command: `man deselectBusGuide`.

`-layer {`*id* | *id1*:*id2*`}`

Specifies the layer or layer range from which bus guide segments are deselected.

The command deselects bus guide segments with the exact specified layer information and not a subset of it.

For example, for a bus guide segment with layer constraint `2:5` if you specify `-layer 5`, the segment is not deselected. To deselect this guide segment you need to specify `-layer 2:5` completely.

`-netGroup {`*`netGroup`* `| {`*`list_of_net_groups`*`}}`

Specifies the name of the net group or list of net groups from which the bus guide segments are deselected.

You can also use wildcard (* or ?) for specifying the net group name.

Example

The following command deselects all the bus guide segments associated with the net group `abcBus` that has layer constraint from metal 3 to metal 7:

```
deselectBusGuide -netGroup abcBus -layer 3:7
```

resetBusGuideMultiColors

`resetBusGuideMultiColors`

Clears the highlighted bus guide colors.

Specify `setBusGuideMultiColors` to color the bus guide(s).

Parameters

None.

Related Topic

- "Bus Planning" chapter in the *Encounter User Guide*
  - Highlighting and Dehighlighting the Bus Guide

selectBusGuide

```
selectBusGuide
     [-help]
     [-area x1 y1 x2 y2]
     [-netGroup {netGroup | {list_of_net_groups}}]
     [-layer {id | id1:id2}]
     [-direction {H | V}]
     [-all]
```

Selects a bus guide segment.

Note: If you do not specify any option, by default, the command selects all the bus guide segments.

Use this command after creating a bus guide segment.

Parameters

`-all`

Selects all the bus guide segments in the design.

`-area` *x1 y1 x2 y2*

Specifies the coordinates of the area from which the bus guide segments are selected.

`-direction {H | V}`

Specifies the direction of the bus guide segment to select. This can be `H` (horizontal) or `V` (vertical).

`-help`

Outputs a brief description that includes type and default information for each `selectBusGuide` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man selectBusGuide`.

`-layer {`*id* | *id1*:*id2*`}`

Specifies the metal layer or layer range from which bus guide segments are selected.

The command selects bus guide segments with the exact specified layer information and not a subset of it. For example, for a bus guide segment with layer constraint `2:5` if you specify `-layer 5`, the segment is not selected by the command. To select this guide segment you need to specify `-layer 2:5` completely.

`-netGroup {`*`netGroup`* `|` `{`*`list_of_net_groups`*`}}`

Specifies the name of the net group or list of net groups from which the bus guide segments are selected.

You can also use wildcard (* or ?) for specifying the net group name.

Example

The following command selects all the horizontal (`H`) bus guide segments associated with the net group `group1`:

```
selectBusGuide -netGroup group1 -direction H
```

selectBusGuideSegment

```
selectBusGuideSegment
     [-help]
     [-box x1 y1 x2 y2]
     [-layer {id | id1:id2}]
     [-netGroup {netGroup}]
```

Selects a bus guide segment with its specified bounding box. This command is mainly used for recording the bus guide segment selection operation in the Encounter command file.

Use this command after creating a bus guide segment.

Parameters

`-box` *x1 y1 x2 y2*

Specifies the coordinates of the bus guide box to select.

`-help`

Outputs a brief description that includes type and default information for each `selectBusGuideSegment` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man selectBusGuideSegment`.

`-layer {`*id* | *id1*:*id2*`}`

Specifies layer or layer range from which the bus guide segment is to be selected.

`-netGroup {`*netGroup* | {*list_of_net_groups*}`}`

Specifies the name of the net group or list of net groups from which the bus guide segment is to be selected.

Example

The following command selects the bus guide segment that has bounding box coordinates `x1` = `15150.7`, `y1` = `7061.0`, `x2` = `15450.7`, `y2` = `11230.0`, layer constraint on metal 3, associated with the net group `busGroup`.

```
selectBusGuideSegment -netGroup busGroup -box 15150.7 7061.0 15450.7 11230.0 -layer
3
```

setBusGuideMultiColors

`setBusGuideMultiColors`

Colors all the bus guides.

Specify `resetBusGuideMultiColors` to clear the bus guide colors.

Parameters

None.

Related Topic

- "Bus Planning" chapter in the *Encounter User Guide*
  - Highlighting and Dehighlighting the Bus Guide

2

Clock Mesh Synthesis Commands

addClockMeshLoad

```
addClockMeshLoad
     [-cell cellName]
     [-maxNrLoad num]
```

Inserts loading cells into the final stage global mesh net, to try to make the load density distribution more uniform. The `addClockMeshLoad` command inserts buffers, inverters, or any cell with one input pin and no output pins, as additional loading. Adding loading cells can help improve skew in cases where the skew is caused by uneven loading.

You can define the loading cells to use in the following ways:

- Specify `addClockMeshLoad -cell cellName`
- Specify loading cells in the clock mesh specification file using the `LoadCell` statement:

  ```
  LoadCell
  + cell1
  + cell2
  ...
  ```

- Allow the software to choose from available loading cells in the library

You can use the `addClockMeshLoad` command after the global mesh has been synthesized (`synthesizeClockMesh`) , and before or after the clock mesh has been routed (`routeClockMesh`). Cadence does not recommend using the `addClockMeshLoad` command after `trimClockMesh`, because it might result in bad skew.

Parameters

`-cell cellName`	Specifies the name of the buffer, inverter, or single input cell to use as the loading cell. *Default:* If you do not specify this parameter, the software chooses from the loading cells that are defined in the `LoadCell` statement in the clock mesh specification file. If there is no `LoadCell` statement defined in the file, the software chooses from the available loading cells in the library.
`-maxNrLoad num`	Specifies the maximum number of loading cells that can be inserted.

analyzeClockMesh

```
analyzeClockMesh
     [-help]
     [-force]
     [-invalidate]
```

Refreshes timing information (delay and transition times) for currently loaded clock meshes.

The clock mesh software caches timing information to avoid unnecessary reanalysis. It does not automatically recalculate timing information every time cached information becomes invalid. Rather, the clock mesh software waits until the timing information is needed and then performs the analysis.

Most clock mesh commands that require mesh delay and transition information automatically trigger analysis as needed (for example, reportClockMesh).

One exception is the display redraw function. To avoid excessive redrawing time, if clock mesh timing results are shown in the Encounter display, the clock mesh software does not update the cached timing information during a redraw. Instead, the clock mesh software displays the clock mesh timing in yellow and writes a warning message in the log file noting that the display is no longer current. In this situation, use the `analyzeClockMesh` command to update the timing information manually.

You can use the `analyzeClockMesh` command any time clock mesh data are specified and implemented.

Parameters

`-force`	Forces the `analyzeClockMesh` command to reanalyze mesh timing data, even if the command believes the cached information is up to date. This parameter is useful on rare occasions when the caching mechanism does not notice that data are invalid (for example, due to changes made by other non-clock-mesh commands).
`-help`	Outputs a brief description that includes type and default information for each `analyzeClockMesh` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man analyzeClockMesh`.
`-invalidate`	Manually invalidates the cached clock mesh timing data.

Example

The following command refreshes timing information:

```
analyzeClockMesh
```

changeClockMeshStatus

change, mesh, placement, routing, status

```
changeClockMeshStatus
     [-root]
     [-topChain]
     [-globalMesh]
     [-localTree]
     [-leaf]
     [-placement | -routing]
     [-load]
     {-fix | -unfix}
```

Changes the placement or routing status of clock mesh instances and nets. It does not change the placement status of macros.

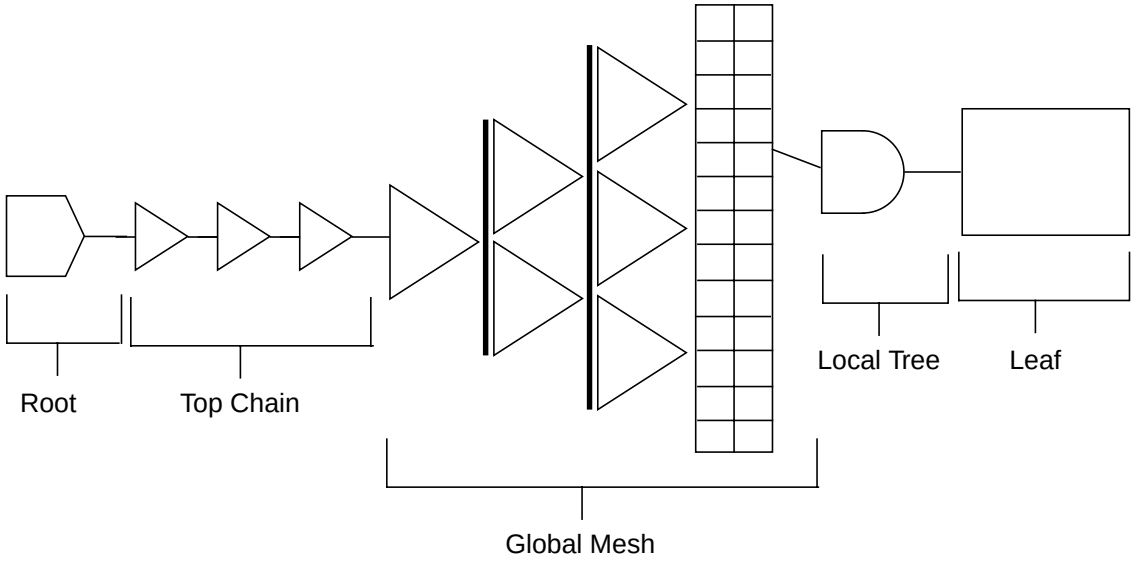
You can use the `changeClockMeshStatus` command before or after synthesizing the clock mesh.

Parameters

`-fix | -unfix`	For placement, changes the status of instances to either `FIXED` or `PLACED`. For routing, changes the status of detailed routes to either `FIXED` or `ROUTED`.
`-globalMesh`	Changes the status of all global mesh pre-drive and final drive buffers, and nets from the first pre-drive net to the final global mesh (up to the inputs of local trees if they exist; otherwise, up to the inputs of leaves).
`-leaf`	Changes the status of leaf instances.
`-load`	Changes the status of the buffers or inverters inserted as loading cells.
`-localTree`	Changes the status of the local tree drivers and the local tree nets (up to the leaf input pins).
`-placement | -routing`	Specifies whether to change the placement or routing status. *Default:* Changes both the placement and routing status

`-root`	Changes the status of the root instance and the net driven by the root (up to the first chain buffer, or the first global mesh pre-drive buffer).
`-topChain`	Changes the status of the chain buffers and the nets they drive.

Figure 2-1 Clock Mesh Structure



Examples

- The following command changes the status of the local trees nets to `ROUTED`:

  ```
  changeClockMeshStatus -localTree -routing -unfix
  ```

- The following command changes the status of all clock mesh nets to `ROUTED`:

  ```
  changeClockMeshStatus -routing -unfix
  ```

- The following command changes the status of all clock mesh instances to `PLACED`, except for global mesh instances and macros:

  ```
  changeClockMeshStatus -placement -unfix
  ```

- The following command changes the status of all instances and wires to `FIXED`, except for macros:

  ```
  changeClockMeshStatus -fix
  ```

characterizeClockMesh

```
characterizeClockMesh
     [-file fileName]
     [-netDelay {port | interconnect}]
```

Analyzes the clock mesh structure and generates an SDF and SDC representation of clock mesh delays and transitions that can be used for static timing analysis by the CTE timing engine.

The `characterizeClockMesh` command generates two files:

- A partial SDF file that contains the cell delays for the mesh drivers, and the interconnect delay for mesh nets.

  Because the generated SDF file does not contain all possible SDF arcs for the multi-drive clock mesh nets, it does not suffer from the memory and performance issues caused by considering all timing arcs. Additionally, there is no loss of accuracy due to dropped timing arcs.

- A pin slew file that contains the input slew for all mesh drivers and the global mesh receivers.

You can load the partial SDF file back into the software for timing analysis using the `read sdf -clock_mesh -persistent` parameters, which instruct the timing system to follow the reduced-arc structure found in the SDF file, instead of following its own reduction process.

The `characterizeClockMesh` command performs analysis and characterization based on delay and transition values derived from the internal analysis methods built into the clock mesh code. This internal analysis can be based on either delay calculation (using the SignalStorm engine), or circuit simulation (loosely integrated UltraSim, or a third-party fast-SPICE simulator). Because the mesh timing analysis can do circuit simulation, it has the potential to be more accurate than relying on the default CTE timing engine behavior, which always uses delay calculation.

Parameters

`-file fileName`
: Generates a partial SDF file (*`fileName`*`.sdf`) and a slew file (*`fileName`*`.sdc`) with the specified name.

  Default: Does not generate the files.

`-netDelay {port | interconnect}`

Controls whether the clock mesh net delays are described using the SDF `PORT` or `INTERCONNECT` delay statements.

Default: `interconnect`

Examples

- The following example command script lists the commands used to perform timing analysis based on precalculated clock mesh delays:

```
setExtractRCMode -specialnet true
specifyClockMesh -file mesh_It.spec
synthesizeClockMesh
routeClockMesh
changeClockMeshStatus -fix
reportClockMesh -delay
characterizeClockMesh -file characterize
loadTimingCon -incr prop.sdc
loadTimingcon -incr characterize.sdc
read_sdf -clock_mesh characterize.sdf -persistent
set_global report_timing_format {instance arc delay slew load annotation}
timeDesign -postCTS
report_timing -to abc/reg_A/D -path_type full_clock -net
```

Related Topcis

- "Working with Clock Mesh Structures" chapter in the *Encounter User Guide*
  - "Analyzing the Clock Mesh"

createClockMeshCutout

```
createClockMeshCutout
     {-inst instanceName | -rect {llx lly urx ury}}
     [-allMacroInst]
     [-instHalo halo]
```

Marks an area that should not be covered by the clock mesh. You can define cutout areas with either an instance name or a location.

Use this command to specify individual clock mesh cutout areas. Specify multiple clock mesh cutout areas with multiple `createClockMeshCutout` commands.

When the clock mesh software encounters a cutout area, the software stops the mesh routing at the cutout boundary, rather than trying to route around the cutout (as in the case of a routing blockage).

This command serves as an alternative to the clock mesh specification file `Cutout` section.

You can use the `createClockMeshCutout` command at any time prior to clock mesh implementation.

Parameters

`-allMacroInst`	Creates cutouts for all macro cell instances.
`-instHalo` *`halo`*	Specifies the distance (in micrometers) to increase the size of instance cutouts from the instance–cell boundary.
`-inst` *`instanceName`*	Defines a cutout area based upon an instance name. **Note:** The cutout is defined by the boundary of a placed instance, with spacing.
`-rect {`*`llx lly urx ury`*`}`	Defines a cutout area based upon a user-defined location, which consists of a set of upper-right and lower-left x and y coordinates. Coordinates are specified in micrometers (μm).

Example

The following commands create clock mesh cutouts using instance names and an explicitly defined location:

```
createClockMeshCutout –rect {-2475 -2463 -2343 -2032}
createClockMeshCutout –inst INST1/C1
```

deleteClockMesh

```
deleteClockMesh
     {-all | -mesh meshList}
     [-loadOnly]
```

Removes the logical and physical implementations of one or more clock meshes.

Note: This command does not affect the clock mesh specification. To remove a clock mesh specification, use the unspecifyClockMesh command.

The deletion process includes the following steps:

- Removing clock mesh routing information (such as special routes and regular routes)
- Removing any clock mesh drivers
- Removing hierarchical driver modules (if present)
- Collapsing the mesh down to a single net

Parameters

`-all`	Deletes all clock meshes.
`-loadOnly`	Deletes only the clock mesh loads, and not other information, such as the clock mesh drivers.
`-mesh` *`meshList`*	Deletes only the specified clock mesh(es). *`meshList`* is a Tcl list of one or more mesh names.

Example

The following command deletes clock meshes for clocks `clk1` and `clk2`:

```
deleteClockMesh -mesh {clk1 clk2}
```

deleteClockMeshCutout

```
deleteClockMeshCutout
     {-all | -inst instanceName | -rect {llx lly urx ury}}
```

Deletes previously defined clock mesh cutout areas.

Parameters

`-all`	Deletes all clock mesh cutout areas.
`-inst` *`instanceName`*	Deletes an existing mesh cutout that has been defined with a matching instance.
`-rect {`*`llx lly urx ury`*`}`	Deletes an existing mesh cutout that has been defined with a matching rectangle.

Example

The following command deletes the instance-based clock mesh cutout `TOP/AI/RAM0`:

```
deleteClockMeshCutout -inst TOP/AI/RAM0
```

deleteClockMeshDriver

```
deleteClockMeshDriver
     [-placeDRV]
     [-missingConnection]
     {list_of_buffers}
```

Deletes mesh driver cells.

Parameters

{*list_of_buffers*}	Specifies a list of driver cells to delete. You can specify wild cards as part of the cell name.
-missingConnection	Removes driver cells that do not have via stacks between their output pins and the clock mesh.
-placeDRV	Deletes driver cells that have placement violations.

displayClockMesh

```
displayClockMesh
     [-mesh meshList]
     [-all]
     [-clear]
     [-colorRange {minps maxps | auto}]
     [-cutout {true | false}]
     [-leafBox {true | false}]
     [-level {leaf | none | n}]
     [-localTree {true | false}]
     [-localTreeStyle {star | route}]
     [-delay]
     [-max]
     [-min]
     [-transition]
     [-edge {rise | fall | trigger}]
```

Controls which clock mesh data are shown in the Encounter display. The following types of clock mesh information can be displayed:

- Mesh cutout areas

  Indicated by yellow-crossed rectangles.

- Mesh leaf bounding box

  Indicated with an unfilled green rectangle.

- Local tree clusters

  Indicated as yellow lines from the local tree buffer to all receivers in the local tree cluster

- Delay or transition times to drivers or mesh leaf points

  Indicated with a rainbow color spectrum to indicate different arrival or transition times at individual drivers or leaves.

Note: If mesh analysis results are not available, use the analyzeClockMesh command to update the display.

Parameters

`-all`	Displays all clock meshes.
`-clear`	Clears the clock mesh display.

`-colorRange {`*`minps maxps`* `| auto}`

Sets the range of timing values associated with the color spectrum.

`auto`	Automatically sets color range to match the current range of delay or transition values.
`minps maxps`	Sets the color scale for the range of specific minimum and maximum values.

`-cutout {true | false}`

Specifies whether to display cutouts.

`-delay` Instructs the clock mesh software to use delay values (as opposed to transition values) when displaying timing data by color.

`-edge {rise | fall | trigger}`

Specifies whether the clock mesh software colors the display based on rise, fall, or trigger edge delay.

Default: `trigger`

`-leafBox {true | false}`

Specifies whether to display the bounding boxes of clock mesh leaves.

`-level {leaf | none |` *`n`*`}`

Controls the display of clock mesh delay or transition times by color.

`leaf`	Displays timing at the clock mesh leaf level.
`none`	Suppresses the display of delay and transition times.
`n`	Displays delay or transition times by level number.

`-localTree {true | false}`

Specifies whether to highlight local tree clusters.

`-localTreeStyle {star | route}]`

	Specifies whether to display the route guide topology (`route`), or the local tree connection flight lines (`star`). This feature can be useful for debugging routing guides, as well as timing. *Default:* `star`
`-max`	Instructs the clock mesh software to use maximum values when displaying timing data by color.
`-mesh` *`meshList`*	Displays only the specified clock mesh(es). *`meshList`* specifies a Tcl list of one or more mesh names.
`-min`	Instructs the clock mesh software to use minimum values when displaying timing data by color.
`-transition`	Instructs the clock mesh software to use transition time values (as opposed to delay values) when displaying timing data by color.

Examples

- The following command displays delay values for level 2 of the mesh `CLK`:

```
displayClockMesh -mesh CLK -delay -level 2
```

getClockMeshMode

```
getClockMeshMode
     [-allowDriverBonding]
     [-allowGlobalBonding]
     [-allowReceiverGrouping]
     [-maxPinMatrixSize]
     [-maxSkewSlewRatio]
     [-optDelayEdge]
     [-optPrefRouteTopology]
     [-optSwapCell]
     [-optSwapPhysEqCellOnly]
     [-optVerbose]
     [-postRouteAnalysis]
     [-preRouteAnalysis]
     [-propagationMode]
     [-quiet]
     [-reduceMeas]
     [-reduceMeasErrTol]
     [-simInputWaveformShape]
     [-synthSearchRegionDiagLevel]
     [-synthSearchStackDiagLevel]
     [-synthVerbose]
     [-uSimPostL]
     [-uSimSimMode]
     [-uSimSpeed]
```

Displays the following information about a specified clock mesh mode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the clock mesh mode parameters.

Parameters

-parameter_names

	Displays information for the specified parameters. You can specify one or more parameters. See `setClockMeshMode` for descriptions of the clock mesh mode parameters you can specify.
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setClockMeshMode` parameters in Tcl list format.

Examples

- The following command displays the current setting for the `-propagationMode` parameter:

  ```
  getClockMeshMode -propagationMode
  ```

  The software displays the following information:

  ```
  -propagationMode max             # enums={min max min_max}, default=max
  max
  ```

- The following command displays the current setting for the `-propagationMode` parameter in Tcl list format:

  ```
  getClockMeshMode -propagationMode -quiet
  ```

  The software displays the following information:

  ```
  max
  ```

- The following command displays the current settings for the `-postRouteAnalysis` and `-propagationMode` parameters:

  ```
  getClockMeshMode -propagationMode -postRouteAnalysis
  ```

  The software displays the following information:

  ```
  -propagationMode max             # enums={min max min_max}, default=max
  -postRouteAnalysis SignalStorm   # enums={SignalStorm UltraSim},
  default=SignalStorm
  {propagationMode max} {postRouteAnalysis SignalStorm}
  ```

- The following command displays the current settings for the `-postRouteAnalysis` and `-propagationMode` parameters in Tcl list format:

  ```
  getClockMeshMode -propagationMode -postRouteAnalysis -quiet
  ```

  The software displays the following information:

  ```
  {propagationMode max} {postRouteAnalysis SignalStorm}
  ```

getClockMeshNets

```
getClockMeshNets
     [-mesh meshList]
```

Returns a Tcl list of net names that are part of the specified mesh(es).

Parameters

`-mesh` *`meshList`*	Returns nets for the specified clock mesh(es). *`meshList`* specifies a Tcl list of one or more mesh names.

Example

The following command returns a Tcl list of net names:

```
getClockMeshNets
mesh/L0_N0 mesh/L1_N0 mesh/L2_N0 mesh/L3_N0
```

optimizeClockMesh

`optimizeClockMesh`

Optimizes local tree timingafter local trees have been synthesized.

The `optimizeClockMesh` command can be used with or without having the global mesh implemented. If the global mesh exists, the command uses the actual global mesh skew (the arrivals at the local tree roots), and transitions at the local tree root inputs to drive optimization. If the global mesh does not exist, the command assumes uniform delay (zero skew) and a uniform transition time.

The `optimizeClockMesh` command can use two methods for optimization:

- Swapping local root cells (that is, cell sizing or swapping)
- Changing the planned local net routing topology. This can be done only for pre-route optimization (that is, before using the `routeClockMesh` command).

You can set clock mesh optimization global parameters with the `setClockMeshMode` command. Optimization parameters set with the `setClockMeshMode` command are then used by the `optimizeClockMesh` command every time it is run.

You can use the `optimizeClockMesh` command before or after routing the clock mesh. However, cell swapping only is available after routing the clock mesh.

Parameters

None

releaseClockMeshResources

`releaseClockMeshResources`

Releases computing resources held by the clock mesh software.

When you use this command, system memory is freed, which deletes the current clock mesh information (for example, clock mesh specifications; cached timing data).

You can use the `releaseClockMeshResources` command whenever a clock mesh is specified.

Parameters

None

Example

The following command releases clock mesh resources:

```
releaseClockMeshResources
```

reportClockMesh

```
reportClockMesh
     [-structure]
     [-connectivity]
     [-delay_summary]
     [-delay]
     [-delay_detail]
     [-power]
     [-power_detail]
     [-macromodel macromodelFile]
     [-latency latencyFile]
     [-signalStormSlew slewFile]
     [-structure | -structure_detail]
     [-mesh meshName]
     [-out reportName]
```

Generates various clock mesh reports.

Parameters

`-connectivity` Specifies that net routing connectivity information be included in the report file or displayed in the Encounter console.

`-delay` Reports the following delay information a report file and the Encounter console:

- A summary table showing the delay, skew, and transition time
- A breakdown of the delay, skew, and transition data for each mesh level

`-delay_detail` Reports additional delay information in a report file and the Encounter console. The information includes:

- A summary table showing delay, skew, and transition data for the whole clock structure
- A breakdown of delay, skew, and transition data for each mesh level
- A report of the delay and transition information for each buffer and mesh leaf

`-delay_summary` Specifies that the delay summary information be included in a report file or displayed in the Encounter console, including:

- A summary table showing delay, skew, and transition data for the whole clock structure

`-latency` *`latencyFile`*

Generates a timing constraint file containing `set_clock_latency` and `set_input_transition` statements.

`-macroModel` *`macromodelFile`*

Generates a file containing clock tree specification file `MacroModel` statements that can be used with clock tree synthesis (CTS).

`-mesh` *`meshName`* Generates clock mesh reports for the specified clock mesh only.

Default: Generates reports for all clock meshes

`-out` *`reportName`* Specifies the output report file. If you specify this parameter, the information in the report appears *only* in the report, not in the Encounter console or Encounter log file.

Default: Report information appears *only* in the Encounter console or Encounter log file.

`-power` Reports total and per-level power information.

`-power_detail` Reports total power information on a per-level, per-net basis.

`-signalStormSlew` *`slewFile`*

Generates a SignalStorm® nanometer delay calculator command file comprising a set of `set_slew` commands for each leaf, describing the transition times for the clock mesh(es). Use the slew file with standalone SignalStorm software (that is, outside the Encounter environment).

`-structure`

Creates a level-based report of the logical clock mesh structure that includes the following information:

- clock root pin
- level number
- fanout information (including number of leaf pins and gated cells)

You can generate a level-based report before or after clock mesh synthesis.

`-structure_detail`

Creates a detailed level-based report of the logical clock mesh structure that includes the following information in addition to that in the `-structure` report:

- clock root pin
- level number
- fanout information (including number of leaf pins and gated cells)
- `FIXED` and `dontTouch` attributes (shown with instance pin name)
- leaf pin names and their rising or falling edge status
- whether clock is being inverted

You can generate a detailed level-based report before or after clock mesh synthesis.

Examples

- The following command generates a level-based report of the logical structure of the H-tree clock mesh `clk`:

  ```
  reportClockMesh -structure
  ```

  The software generates the following report:

  ```
  Structure Section


  clk (root)
      Level: L0 (root)
      Net: clk
      Nr. Buffer: 1
  ```

```
    mesh_module/L01_B000 (A->Y) (buffer)

      Level: L1 (1st-level pre-drive)
      Net: mesh_module/L01_N0
      Nr. Buffer: 2

      mesh_module/L02_B000 (A->Y) (buffer)

        Level: L2 (2nd-level pre-drive)
        Net: mesh_module/L02_N0
        Nr. Buffer: 2

        mesh_module/L03_B000, ... (A->Y) (2 buffers in parallel)
         (see below Level: L3 for output net mesh_module_L03_N0)

      mesh_module/L02_B001 (A->Y) (buffer)

        Level: L2 (2nd-level pre-drive)
        Net: mesh_module/L02_N1
        Nr. Buffer: 2

        mesh_module/L03_B000, ... (A->Y) (2 buffers in parallel)
          (driven from mesh_module/L02_N0)
        mesh_module/L03_B002, ... (A->Y) (2 buffers in parallel)
          (driven from mesh_module/L02_N1)

           Level: L3 (mesh drive)
           Net: mesh_module_L03_N0
           Nr. Leaf: 807

    ...
```

- The following command generates a detailed level-based report for the same H-tree clock mesh:

  ```
  reportClockMesh -structure_detail
  ```

  The software generates the following report:

  ```
  Structure Section


  clk (root) (fixed)
      Level: L0 (root)
      Net: clk
      Nr. Buffer: 1

      mesh_module/L01_B000 (A->Y) (buffer) (fixed)

        Level: L1 (1st-level pre-drive)
        Net: mesh_module/L01_N0
        Nr. Buffer: 2

        mesh_module/L02_B000 (A->Y) CLKBUFX20 (buffer) (fixed)

          Level: L2 (2nd-level pre-drive)
          Net: mesh_module/L02_N0
          Nr. Buffer: 2
  ```

```
    mesh_module/L03_B000 (A->Y) CLKBUFX20 (buffer) (fixed)
    mesh_module/L03_B001 (A->Y) CLKBUFX20 (buffer) (fixed)
      (see below Level: L3 for output net mesh_module_L03_N0)

  mesh_module/L02_B001 (A->Y) CLKBUFX20 (buffer) (fixed)


    Level: L2 (2nd-level pre-drive)
    Net: mesh_module/L02_N1
    Nr. Buffer: 2

    mesh_module/L03_B000 (A->Y) CLKBUFX20 (buffer) (fixed)
      (driven from mesh_module/L02_N0)
    mesh_module/L03_B001 (A->Y) CLKBUFX20 (buffer) (fixed)
      (driven from mesh_module/L02_N0)
    mesh_module/L03_B002 (A->Y) CLKBUFX20 (buffer) (fixed)
      (driven from mesh_module/L02_N1)
    mesh_module/L03_B003 (A->Y) CLKBUFX20 (buffer) (fixed)
      (driven from mesh_module/L02_N1)

       Level: L3 (mesh drive)
       Net: mesh_module_L03_N0
       Nr. Leaf: 807

       atahost1/dma_dev0_tm_reg_0/CK SDFFRX4 (leaf) (rising)
       atahost1/dma_dev0_tm_reg_1/CK SDFFRX4 (leaf) (rising)
       atahost1/dma_dev0_tm_reg_2/CK SDFFSX4 (leaf) (rising)
       atahost1/dma_dev0_tm_reg_3/CK SDFFRX4 (leaf) (rising)
       atahost1/dma_dev0_tm_reg_4/CK SDFFRX4 (leaf) (rising)
       atahost1/dma_dev0_tm_reg_5/CK SDFFRX4 (leaf) (rising)
       atahost1/dma_dev0_tm_reg_6/CK SDFFRX4 (leaf) (rising)
       atahost1/dma_dev0_tm_reg_7/CK SDFFRX4 (leaf) (rising)
...
```

reportClockMeshPath

```
reportClockMeshPath
     [-pathType {min | max}]
     [-edge {rise | fall | trigger}]
     [-out fileName]
     -to pinName
```

Reports the cell delay, net delay, and loading condition from the clock root pin to the specified pin.

Parameters

`-edge {rise | fall | trigger}`

Reports the rise, fall, or trigger edge delay for the pin.

Default: `trigger` (for leaf pin), `rise` (for non-leaf pin)

`-out` *`fileName`*

Writes the report to the specified file.

Default: Prints out the report in the log file

`-pathType {min | max}`

Reports path information for either the path with the longest delay (`max`), or the path with the shortest delay (`min`).

Default: Reports information for the path with the longest delay.

`-to` *`pinName`*

Reports the path information from the clock source to the specified pin.

Examples

- The following command reports the path information from the clock root pin to the pin `F10K/D2/C1/X1/FF5/CKN`:

  ```
  reportClockMeshPath -to F10K/D2/C1/X1/FF5/CKN
  ```

  The software generates the following report:

  ```
  Endpoint: FABC/neg_reg/CKN (falling edge)
  Latency: 0.4986 ns

  Object               R/F   Cell         Load    Slew    Delay   Arrival
  ------------------------------------------------------------------------
  clk10K                f                0.021   0.1100  0.0000  0.0000
  big_clk/L01_B000/A    f    CLKBUFX64           0.1258  0.0478  0.0478
  ```

```
big_clk/L01_B000/Y    f    CLKBUFX64    0.083   0.1041  0.1224  0.1702
big_clk/L02_B002/A    f    CLKBUFX64            0.1046  0.0021  0.1723
big_clk/L02_B002/Y    f    CLKBUFX64    0.084   0.0897  0.1111  0.2834
big_clk/L03_B008/A    f    CLKBUFX64            0.0899  0.0009  0.2843
big_clk/L03_B008/Y    f    CLKBUFX64    0.248   0.3341  0.2071  0.4919
FABC/neg_ref/CKN      f    SDFFXL               0.3344  0.0072  0.4986
```

routeClockMesh

```
routeClockMesh
     [-help]
     [-cpu number]
     [-mesh meshList]
     [-noRouteGuide]
```

Completes the detailed routing for one or more clock meshes.

Clock mesh routing is created by two different steps that use two different types of routing:

- `synthesizeClockMesh` creates “special net” routes for mesh trunks and branches.
- `routeClockMesh` starts the integrated Nanoroute router to complete the detailed routing for any remaining clock mesh connections.

The clock mesh software uses detailed routing for the following types of mesh connections:

- Nets in the top-level cascade chain between the clock root and the inputs of the first-level mesh drivers are routed entirely with regular detailed routes. These nets include the root net and nets driven by the outputs of any top-level cascade chain buffers.
- Connections between mesh driver inputs and the special routing in the previous level’s predrive net.
- Connections between the final-level mesh special routes and either
  - The leaf inputs or
  - The local buffer inputs (if the mesh includes local trees).
- Routing between local buffer outputs and mesh leaves (if the mesh includes local trees).

Note: By default, the `routeClockMesh` command generates a route guide before calling NanoRoute to route the local tree nets.

You can use the `routeClockMesh` command after clock mesh structures have been synthesized.

Parameters

`-cpu` *`number`*	Temporarily overrides the NanoRoute router’s `setNanoRouteMode -envNumberProcessor` parameter, which specifies the number of processors to use (on one workstation) for multi-threading.

`-help`	Outputs a brief description that includes type and default information for each `routeClockMesh` parameter. For a detailed description of the command and all of its parameters, use the man command: `man routeClockMesh`.
`-mesh` *`meshList`*	Instructs the NanoRoute router to route only the specified clock mesh(es), if multiple meshes exist in a design. *`meshList`* specifies a Tcl list of one or more mesh names.
`-noRouteGuide`	Does not create a route guide before calling NanoRoute to route local clock tree nets.

Examples

- The following command specifies that two processors should be used when the NanoRoute router routes meshes `mesh1` and `mesh2`:

```
routeClockMesh -cpu 2 -mesh {mesh1 mesh2}
```

saveClockMeshSpec

```
saveClockMeshSpec
     [-help]
     -file fileName
```

Saves the current clock mesh specification (which is in system memory) to a file.

Parameters

`-file fileName`	Specifies the name of a new or revised clock mesh specification file.
`-help`	Outputs a brief description that includes type and default information for each `saveClockMeshSpec` parameter. For a detailed description of the command and all of its parameters, use the man command: `man saveClockMeshSpec`.

Example

The following command saves the clock mesh specification to the file `mymesh.spec`:

```
specifyClockMesh -file mymesh.spec
```

setClockMeshMode

```
setClockMeshMode
     [-help]
     [-reset]
     [-allowDriverBonding {true | false}]
     [-allowGlobalBonding {true | false}]
     [-allowReceiverGrouping {true | false}]
     [-maxPinMatrixSize n]
     [-maxSkewSlewRatio ratio]
     [-optDelayEdge {rise | fall | trigger}]
     [-optPrefRouteTopology {true | false}]
     [-optSwapCell {true | false}]
     [-optSwapPhysEqCellOnly {true | false}]
     [-optVerbose {true | false}]
     [-postRouteAnalysis {SignalStorm | UltraSim}]
     [-preRouteAnalysis {SignalStorm | UltraSim}]
     [-propagationMode {min | max | min_max}]
     [-reduceMeas {true | false}]
     [-reduceMeasErrTol ps]
     [-simInputWaveformShape {ramp | exponential}]
     [-synthSearchRegionDiagLevel {None | LastError | AllError | All}]
     [-synthSearchStackDiagLevel {None | LastError}]
     [-synthVerbose {true | false}]
     [-uSimPostL {0 | 1 | 2 | 3 | 4}]
     [-uSimSimMode {DF | DA | MS | A | S}]
     [-uSimSpeed {1 | 2 | 3 | 4 | 5 | 6 | 7 | 8}]
```

Sets global parameters for clock mesh synthesis, optimization, and analysis. Parameters you specify with the `setClockMeshMode` command are then used automatically whenever you run the `synthesizeClockMesh`, `optimizeClockMesh`, and `analyzeClockMesh` commands.

Use `getClockMeshMode` to display the current settings for the `setClockMeshMode` command.

Parameters

`-allowDriverBonding {true | false}`

Specifies whether to allow local-driver bonding during mesh delay calculation.

Default: `true`

`-allowGlobalBonding {true | false}`

Specifies whether clock mesh delay calculation should combine all mesh drivers together into a single large buffer.

Global bonding can improve performance and memory usage for delay calculation for large mesh structures, but the slew results will likely be overly optimistic.

Default: `false`

`-allowReceiverGrouping {true | false}`

Specifies whether to allow receiver grouping during mesh delay calculation.

Default: `true`

`-help`

Outputs a brief description that includes type and default information for each `setClockMeshMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setClockMeshMode`.

`-maxPinMatrixSize` *`n`*

Specifies a limit on the driver–receiver product before local driver bonding or receiver grouping is triggered. Specify a value of `0` for unlimited pin matrix size.

Default: `500000`

`-maxSkewSlewRatio` *`ratio`*

Specifies a ratio of input skew to input slew (for parallel net drivers) above which the clock mesh software issues a warning.

Default: `0.5`

`-optDelayEdge {trigger | rise | fall}`

Specifies which edge to use for optimizing skew.

Default: `trigger`

`trigger`	Minimizes skew based on the trigger edge of the flops.
`rise`	Minimizes skew based on the rising edge of the flops.

`fall`	Minimizes skew based on the falling edge of the flops.

`-optPrefRouteTopology {true | false}`

Specifies whether to change the routing topology for local tree nets, in order to improve skew and transition.

Note: This parameter has no effect if the local trees have already been routed.

Default: `true`

`-optSwapCell {true | false}`

Specifies whether to enable cell swapping.

Default: `true`

`-optSwapPhysEqCellOnly {true | false}`

Specifies whether to restrict cell swapping to only cells that are physically equivalent (that is, the same size, pin geometry, and obstruction geometry). This feature can be useful during post-route optimization because it restricts the cell changes to those that do not require ECO place-and-route.

Default: `false`

`-optVerbose {true | false}`

Specifies whether to generate more verbose messages during optimization.

Default: `false`

`-postRouteAnalysis {SignalStorm | UltraSim}`

Specifries whether the clock mesh software uses integrated SignalStorm or the Virtuoso® UltraSim™ fast SPICE simulator for postroute mesh analysis.

Default: `SignalStorm`

`-preRouteAnalysis {SignalStorm | UltraSim}`

Specifies whether the clock mesh software uses integrated SignalStorm or the Virtuoso® UltraSim™ simulator for preroute mesh analysis.

Default: `SignalStorm`

`-propagationMode {min | max | min_max}`

Controls how clock mesh analysis propagates timing through a net with multiple drivers.

Default: `max`

`min`	Specifies that each driver should take the minimum value for delay and transition among all driver inputs.
`min_max`	Specifies that each driver receives its actual transition and delay values. Skew at the net drivers is propagated through the net to the net receivers, causing skew uncertainty to accumulate stage by stage. Therefore, the results from delay analysis might be overly pessimistic.
`max`	Specifies that each driver should take the maximum value for delay and transition among all driver inputs.

`-reduceMeas {true | false}`

Specifies whether to eleminate SPICE measure statements for nodes that are likely to have timing that is very similar to other nodes.

Default: `false`

`-reduceMeasErrTol` *`picoseconds`*

Specifies the threshold that controls which measure statements can be eliminated if `-reduceMeas` is `true`. This value is an estimate.

Default: `0.5`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setClockMeshMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-simInputWaveformShape {ramp | exponential}`

Controls the shape of the waveform provided as stimulus at the clock mesh root during SPICE netlist generation (and simulation).

Regardless of the shape, the waveform parameters are chosen so that the measured rise and fall times (between appropriate measuring voltages) match the specified root transition time.

The input waveform shape can be expected to have some impact on the cell delay (and to a lesser extent, output transition time) of the first logic level of the clock mesh (that is, the root cell, the first top-chain buffer, or the first pre-drive stage), but negligible impact overall. In terms of overall mesh timing, for most cases, the input waveform shape will have a small impact on total latency, and little or no impact on final skew.

Default: `ramp`

`ramp`	Generates a simple piece-wise linear ramp shape, using PWL function.
`exponential`	Generates an exponential waveform, using EXP function.

`-synthSearchRegionDiagLevel {None | LastError | AllError | All}`

Enables the temporary display of region markers in the Encounter display for various wire and placement checks during clock mesh synthesis.

Default: `LastError`

`All`	Displays regions for all searches (successful and failing).
`AllError`	Displays regions for all failing searches.
`LastError`	Displays regions for the last failing search.
`None`	No search regions are displayed.

`-synthSearchStackDiagLevel {None | LastError}`

Specifies whether to create diagnostic stack information for failures during clock mesh synthesis.

Default: `LastError`

`LastError`	Displays diagnostic stack information for the last failing synthesis step.
`None`	No diagnostic stack information is provided.

`-synthVerbose {true | false}`

Specifies whether to generate more verbose messages during clock mesh synthesis.

Default: `false`

`-uSimPostL` *`0-4`* Specifies the Virtuoso® UltraSim™ simulator post-layout level.

Default: `1`

`-uSimSimMode {DF | DA | MS | A | S}`

Specifies the Virtuoso UltraSim simulator mode parameters that control simulation performance and accuracy.

See the *Virtuoso® UltraSim User Guide* for details on simulation modes.

Default: `MS`

`DF`	Specifies the *digital fast* simulation mode.
`DA`	Specifies the *digital accurate* simulation mode.
`MS`	Specifies the *mixed-signal* simulation mode.
`A`	Specifies the *analog* simulation mode.
`S`	Specifies the *SPICE* simulation mode.

`-uSimSpeed` *`1-8`* Specifies a Virtuoso UltraSim simulator speed/accuracy setting.

A setting of `1` represents the slowest simulation speed but produces the most accurate result.

A value of `8` represents the fastest simulation speed but produces the least accurate results.

Default: `5`

Examples

- The following command instructs the software to use the Virtuoso UltraSim simulator for postroute mesh analysis:

  ```
  setClockMeshMode -postRouteAnalysis UltraSim
  ```

- The following command instructs the software to perform cell swapping during clock mesh optimization:

  ```
  setClockMeshMode -optSwapCell true
  ```

- The following command resets the `-postRouteAnalysis` parameter to its default value:

  ```
  setClockMeshMode -reset -postRouteAnalysis
  ```

- The following command resets all of the `setClockMeshMode` parameters to their default values:

  ```
  setClockMeshMode -reset
  ```

specifyClockMesh

```
specifyClockMesh
     [-help]
     -file fileName
     [-trace fileName]
```

Loads a clock mesh specification file into system memory.

You can use this command repeatedly to change the current clock mesh specification. If you specify a clock mesh specification file that you have already loaded in your current Encounter session, that specification is updated in your system's memory.

If any scope information (such as root pin or leaf pin information) is set in the clock mesh specification file, the scope is assumed to be completely defined. Any change to scope causes the clock mesh software to retrace the clock mesh. Retracing can cause the loss of some computed clock mesh information (for example, delay information).

Note: *Scope* is the logical extent of the clock domain from the clock root, through any clock mesh drivers, to the leaves.

Parameters

`-file fileName`	Specifies the name of the clock mesh specification file to be loaded.
`-help`	Outputs a brief description that includes type and default information for each `specifyClockMesh` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man specifyClockMesh`.
`-trace fileName`	Creates a detailed netlist trace log file. You can use this file to help debug errors in the clock mesh specification.

Example

- The following command loads the clock mesh specification file `mesh1` into system memory:

  ```
  specifyClockMesh -file mesh1
  ```

Related Topcis

- "Clock Mesh Specification File" chapter in the *Encounter User Guide*

spiceClockMesh

```
spiceClockMesh
     [-mesh meshList]
     [-nosubckt]
     [-nocdb]
     [-reduceMeas tolerance]
     [-ultraSim]
     [-readMapMeas {map meas}]
     [-gzip]
```

Creates a SPICE netlist to simulate one or more clock meshes.

For each clock mesh, the `spiceClockMesh` command creates two files:

- *`meshName`*`.sp` (the SPICE netlist)
- *`meshName`*`.mp` (a node name-mapping file)

You can use the `spiceClockMesh` command whenever a clock mesh is specified and implemented.

Parameters

`-gzip`
: Creates compressed netlist and mapping files.

  Default: If you do not specify this parameter, the clock mesh software creates a SPICE netlist and mapping file in plain text.

`-mesh` *`meshList`*
: Specifies a clock mesh for which you want to create SPICE output files.

  `meshList` specifies a Tcl list of one or more mesh names.

`-nocdb`
: Instructs the clock mesh software not to read driver subcircuit and model information from the cdb database file.

`-nosubckt`
: Instructs the clock mesh software not to write subcircuit definitions or model information to the SPICE deck.

`-readMapMeas {`*`map meas`*`}`
: Instructs the clock mesh software to load a mapping file and a measurement file.

  Use this parameter for backannotation when you use an external SPICE simulator run.

  You must specify a two-element Tcl list with these values:

map	The name of a mapping file.
meas	The name of a measurement file.

`-reduceMeas` *tolerance*

Specifies a tolerance for measure statement redirection.

If you do not specify this parameter, the clock mesh software makes no attempt to reduce measure statements.

`-ultraSim` Formats the netlist to be compatible with the Virtuoso UltraSim simulator.

Examples

- The following command creates a SPICE netlist for an external simulator run:

  ```
  spiceClockMesh -mesh sys_clk
  ```

- The following command backannotates the simulation results:

  ```
  spiceClockMesh -readMapMeas {sys_clk.map sys_clk.mt0}
  ```

swapClockMeshDriver

```
swapClockMeshDriver
     [-ignoreFootprint]
     -cell cellName buffer_names
```

Swaps clock mesh driver cells or swaps between true mesh drivers and dummy drivers.

A *dummy driver* is a cell is similar in pin-out to a buffer—but without any driveability. A dummy driver has the following characteristics:

- It has exactly one input.
- It has exactly one output.
- There is no timing arc between the input and output.

The new cell must meet the following requirements to be considered a replacement cell:

- The new cell must have a physical footprint that is compatible with the original cell (unless you specify `-ignoreFootprint`). (Compatibility is defined by cell size, pin geometry, and obstruction geometry.)
- The polarity of the new cell must be compatible with the polarity of the original cell.

  The following types of replacement satisfy the polarity requirement:

  - A buffer can replace another buffer.
  - An inverter can replace another inverter.
  - A buffer or inverter can replace a dummy driver as long as the new cell will not cause a conflict in path polarity at the instance's output net. In practice this simply means that you cannot switch between a buffer and inverter by going through a dummy buffer as an intermediate step.
  - A dummy driver replace either a buffer or inverter as long as the replacement does not deprive the instance's output net of any true drivers.

The following table lists allowable swaps.

	Original Cell		
New Cell	**Buffer**	**Inverter**	**Dummy**
Buffer	Yes	No	Yes (see Note 1)
Inverter	No	Yes	Yes (see Note 1)
Dummy	Yes (see Note 2)	Yes (see Note 2)	Yes

1. This swap is allowed as long as the swap does not cause a conflict in path polarity.
2. This swap is allowed as long as the swap does not deprive a net of any true driver.

You can use the `swapClockMeshDriver` command any time a mesh with drivers is specified and implemented. The overall goal of this command is mesh driver optimization.

Parameters

`buffer_names`	Lists instance names or wildcard patterns (`*` or `?`) that identify buffers whose cells can be swapped.
`-cell` *`cellName`*	Specifies the new cell for which to swap the driver cell.
`-ignoreFootprint`	Relaxes the requirement that the new cell have a physical footprint that is compatible with the physical footprint of the original cell.

Example

The following command swaps any buffer `CLK/L1_B2*` with the new cell `CLKBUFX20`:

```
swapClockMeshDriver -cell CLKBUFX20 CLK/L1_B2*
```

synthesizeClockMesh

```
synthesizeClockMesh
     [-mesh meshList]
     [-unfix]
     [-topChain]
     [-globalMesh]
     [-localTree]
```

Synthesizes (implements) one or more clock meshes. Synthesis consists of updating the netlist to include mesh drivers, placing the drivers, an creating mesh special routes.

It is possible to repeatedly synthesize the same mesh, without deleting it beforehand, to experiment with different specification parameters or settings.

The `synthesizeClockMesh` command can work independently on the top-level buffer chain, the global mesh structure, and the local tree level. This allows you to adjust and resynthesize a single part of the mesh structure without having to repeatedly implement the entire mesh. In many cases, the clock mesh software can work with a partial mesh structure. For example, it is possible to create local trees for a manually created global mesh.

Note: If you do not specify any of the `-topChain`, `-globalMesh`, or `-localTree` parameters, the clock mesh software synthesizes all those sections of the clock mesh.

You should perform clock mesh synthesis after placement, but before detailed routing.

Parameters

`-globalMesh`	Synthesizes the global mesh.
`-localTree`	Synthesizes the local trees, if they are defined and enabled in the clock tree specification file.
`-mesh` *`meshList`*	Instructs the clock mesh software to synthesize only the specified clock mesh(es). *`meshList`* specifies a Tcl list of one or more mesh names.
`-topChain`	Synthesizes the top-level chain, if one is defined and enabled in the clock mesh specification file.
`-unfix`	Unfixes any fixed drivers before synthesizing the clock mesh(es). **Note:** If your design contains clock meshes that have fixed buffers and if you do not use this parameter, the command will fail.

Example

The following command synthesizes clock meshes for local trees only:

```
synthesizeClockMesh -localTree
```

trimClockMesh

```
trimClockMesh
     [-mesh meshList]
     [-deleteNoLoadBranches]
     [-deleteNoLoadBranchSections]
     [-trimWireEnds]
     [-layers layerList]
```

Removes unused metal from the clock mesh. The `trimClockMesh` command trims trunks and branches so they do not extend beyond the last routing connection. Trimming reduces capacitance and prevents antenna violations.

In some situations, clock mesh branches that are not connected to any loads can exist in sparse areas of the design. If you want such “no load” branches to be removed, use the `-deleteNoLoadBranches` parameter.

If you do not specify any of the `-trimWireEnds`, `-deleteNoLoadBranches`, or `-deleteNoLoadBranchSections` parameters, the clock mesh software performs wire end trimming by default.

Use the `trimClockMesh` command only after mesh detailed routing is complete.

Parameters

`-deleteNoLoadBranches`

Removes (entire) branches if the (entire) branch does not connect to any loads.

`-deleteNoLoadBranchSections`

Removes portions of branches (sections between trunks) that do not connect to any loads.

`-layers` *`layerList`*
: Limits clock mesh trimming to specific layers. Use LEF layer names with this parameter.

  Default: If you do not specify this parameter, all layers are considered for clock mesh trimming.

`-mesh` *`meshList`*
: Specifies the clock mesh(es) to be trimmed.

  `meshList` specifies a Tcl list of one or more mesh names.

`-trimWireEnds`
: Trims back trunks and branches from their endpoints to the first routing connection or via point.

Example

The following command trims trunks and branches for layer `METAL7`:

```
trimClockMesh -layer METAL7 -trimWireEnds
```

unspecifyClockMesh

```
unspecifyClockMesh
     [-all]
     [-mesh meshList]
```

Removes the specification for one or more clock meshes.

Unspecifying a clock mesh does not change the design database (for example, netlist contents or routing). This command only removes the currently loaded specification from memory.

Note: Use the displayClockMesh command to remove mesh implementations.

Parameters

`-all`	Removes *all* mesh specification information from system memory, including cutouts and route type definitions.
`-mesh meshList`	Removes mesh-specific information from system memory. `meshList` specifies a Tcl list of one or more mesh names.

Example

The following command removes all mesh specification information from system memory:

```
unspecifyClockMesh -all
```

3

Clock Tree Synthesis Commands

addCTSCellList

```
addCTSCellList
     {cell1 [cell2...]}
```

Specifies the list of buffers and inverters to use during clock tree synthesis. You must specify atleast one cell.

Parameters

{*cell1* [*cell2...*]}	Specifies the list of buffers and inverters to use during clock tree synthesis.

Example

- The following command instructs the software to use the `INV1` and `BUF2` cells during clock tree synthesis:

  ```
  addCTSCellList   INV1 BUF2
  ```

analyzeClockTreeSpec

```
analyzeClockTreeSpec
     [-honorMaxDelay value]
     [-genSDCOnly fileName]
```

Analyzes the loaded clock tree specification file, runs timing analysis, and generates a timing report that can be used to assess whether a clock tree specification file is reasonably correct before actually running clock tree synthesis.

When specified, the software traces the clock paths based on the clock tree specification file, and generates an SDC constraints file that contains `set_clock_latency` assertions with appropriate latency values for all input pins, such as leaf pins, and macro models. The software then loads the generated SDC file and performs timing analysis.

You can analyze the resulting timing report for large jumps in total negative slack (TNS) and worst negative slack (WNS), which can indicate that something is missing or incorrectly specified in the clock tree specification file.

The `analyzeClockTreeSpec` command can be used to check:

- The effectiveness of excluded pins
- Whether there is a need to group certain clocks
- Whether there are clock divider circuits that need to be balanced separately.
- Whether there are flops being left out during clock tree synthesis (that is, there are no `AutoCTSRootPin` statements declared to drive them).

The software uses the following rules for applying insertion delay values:

- Cross-over clocks are assigned the same insertion delay value.
- Groups of leaf pins driven by a clock (specified by a `ClkGroup` statement in the specification file) are assigned the same insertion delay value.
- Leaf pins of clocks that are not cross-overs, or are not grouped as a `clkGroup` are assigned different non-zero insertion delay values.
- Through pins and excluded pins are assigned an insertion delay value of `0`.
- Macro models are assigned an insertion delay value equal to the leaf pin value minus the maximum delay of the macro model.

Important

The following limitations exist for the `analyzeClockTreeSpec` command:

- The timing reports it generates do not match exactly with the actual post-CTS timing report, because the assumption is based on the clock latency value assigned to each input pin, whether it is a leaf pin, excluded pin, or another type of input pin.
- The report does not give recommendations for correcting the clock tree specification file. Use the information in the report to make judgements on how to adjust the specification file.
- The command is not supported for Multi-Mode multi-Corner CTS.

Parameters

`-genSDCOnly` *`fileName`*

Generates an SDC constraints file with the specified name.

Default: *`design`*`.cts2sdc`

`-honorMaxDelay` *`value`*

Uses the specified maximum delay value for the latency value. The `MaxDelay` statement value in the clock tree specification file will be the converted `set_clock_latency` value.

Note: Cross-over clocks will stave have the same insertion value.

Default: The software decides the latency value based on the previously described rules.

changeClockStatus

```
changeClockStatus
     {-clk clkName | -all}
     {[-fixedBuffers | -noFixedBuffers]
      [-fixedNetWires | -noFixedNetWires]
      [-useClock | -noUseClock]
      [-fixedLeafInst | -noFixedLeafInst]
      [-fixedNonLeafInst | -noFixedNonLeafInst]}
```

Changes the placement status of buffers and the routing status of nets based on the current clock tree specification file. For example, you can use this command to “unfix” buffers if you want to delete a clock tree that contains FIXED buffers.

Note: To change the status based on the DEF + USE CLOCK net attribute, use the changeUseClockNetStatus command.

You can use the changeClockStatus command after performing clock tree synthesis on the design.

Parameters

-all	Changes the placement status all clock roots defined in the clock tree specification file.
-clk *clkName*	Specifies the clock root name defined in the clock tree specification file.
-fixedBuffers	Changes the placement status of buffers, inverters, flip-flops, and gating cells from PLACED to FIXED.
-fixedLeafInst	Changes the placement status of leaf instances from PLACED to FIXED.
-fixedNetWires	Changes status of wires in clock nets from ROUTED to FIXED.
-fixedNonLeafInst	Changes the placement status of non-leaf instances from PLACED to FIXED.
-noFixedBuffers	Changes the placement status of buffers, inverters, flip-flops, and gating cells from FIXED to PLACED.
-noFixedLeafInst	Changes the placement status of leaf instances from FIXED to PLACED.
-noFixedNetWires	Changes the status of wires in clock nets from FIXED to ROUTED.

`-noFixedNonLeafInst`	Changes the placement status of non-leaf instances from `FIXED` to `PLACED`.
`-noUseClock`	Instructs CTS to omit `+ USE CLOCK` statements in the output DEF file.
`-useClock`	Instructs CTS to include `+ USE CLOCK` statements in the output DEF file.

Examples

- The following example shows how to change the placement status of `FIXED` buffers in clock tree `clk` so CTS can remove that clock tree:

```
specifyClockTree -clkfile design_32.cts
changeClockStatus -clk clk -noFixedBuffers
deleteClockTree -clk clk
```

- The following command changes the placement status of the buffers in clock tree `MCK_GE` from “placed” to “fixed”:

```
specifyClockTree -clkfile design_32.cts
changeClockStatus -clk MCK_GE -fixedBuffers
```

Related Topics

- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*

changeUseClockNetStatus

```
changeUseClockNetStatus
     {[-fixedNetWires | -noFixedNetWires]
      [-fixedFF | -noFixedFF]}
```

Changes the placement status of flip-flops and the routing status of nets with the DEF `+ USE CLOCK` attribute.

Note: To change the status based on the clock tree specification file, use the `changeClockStatus` command.

You can use the `changeUseClockNetStatus` command after performing clock tree synthesis on the design.

Parameters

`-fixedFF`	Changes the status of flip-flops connected to clock nets to `FIXED`.
`-fixedNetWires`	Changes the status of nets marked `+ USE CLOCK` in the DEF file to `FIXED`.
`-noFixedFF`	Changes the status of flip-flops connected to clock nets from `FIXED` to `PLACED`.
`-noFixedNetWires`	Changes the status of nets marked `+ USE CLOCK` in the DEF file from `FIXED` to `ROUTED`.

checkClockTreeCellHalo

```
checkClockTreeCellHalo
     [-file fileName]
     [-hilite]
```

Verifies if there is any cell halo violation.

Parameters

`-file fileName`	Specifies the output file where cell halo violation is reported.
`-hilite`	Highlights the clock instances that pass or fail the cell halo rule. Green color indicates clock instances that successully pass the cell halo rule. Red color indicates clock instances that fail the cell halo rule.

Examples

- The following command verifies the cell halo violation in the `myfile` file:

  ```
  checkClockTreeCellHalo -file myfile -hilite
  ```

ckCloneGate

```
ckCloneGate
     [-clk clockName]
     [-ignorePreplaced]
     [-ilm]
     [-ignoreDontTouch]
     [-forceReconvergent]
     [-breakLoop]
     [-noUpsizeCell]
     [-msvAware]
     [-specOnly]
     [-check]
     [-icgOnly]
     [-handleInstGrp]
     [-enableStrictCloning]
     [-timingDriven]
     [-file fileName]
```

Clones (or duplicates) clock gating components and redistributes their gated loads, depending on the parameters you specify.

CTS automatically detects integrated clock gating cells and simple combinational gating cells. *Clock gating components* consist of

- Buffers
- Inverters
- AND gates
- OR gates
- Any other logical element (defined in the library) that
  - Appears in the clock tree before clock tree synthesis inserts any buffers or inverters and
  - Drives at least one synchronous pin

The cloning functionality *does not*

- Correct design rule violations on the data path. (For example, cloning a large number of cells can create high-fanout nets for the enable signal. You would need to run in-place optimization (IPO) to correct such problems.)
- Consider gating check timing.

You can use the `ckCloneGate` command after loading a clock tree specification file, and before running `ckSynthesis`. For specification file information, see Creating a Clock Tree Specification File in the “Synthesizing Clock Trees” chapter of the *Encounter User Guide*.

Parameters

`-breakLoop`

Instructs CTS to automatically detect loops in clock trees. If CTS detects a loop in a clock tree, CTS

- Displays a message
- Breaks the loop
- Continues to trace through the clock tree

`-check`

Creates a report showing how CTS will perform cloning.

The report provides the following information:

- Clock name
- Number of master gate instances
- Instance name
- Cell name
- Number of sinks
- Number of clock gating cells
- Number of cloned cells to be added to the tree
- A script that you can add to the clock tree specification file

Note: For combinational clock gating cells, `-check` reports only the AND/OR gates (not latches).

`-clk` *`clockName`*

Specifies the clock root name defined in the clock tree specification file.

Default: If you omit this parameter, `ckCloneGate` uses the names of all the clocks in the specification file.

`-enableStrictCloning`

	Performs cloning based on removing all of the transition violations based on constraints in the clock tree specification file. **Note:** Using this parameter can result in a substantial increase in the number of clock gating cells being cloned. *Default:* Performs cloning based on heuristics that relax the constraints for internal calculations. **Note:** This parameter is not supported with timing-driven cloning.
`-file` *`fileName`*	Saves the information generated by `-check` to the specified file.
`-forceReconvergent`	Forces CTS to trace clock reconvergent paths. If your design contains clock reconvergent paths and you do not specify this parameter, CTS halts and issues an error message.
`-handleInstGrp`	Clones groups of discrete components (such as Latches and AND gates) that are used as clock gating cells. This parameter is useful for designs that do not use integrated clock gating (ICG) cells. The difference between a component group and an ICG cell is that the ICG cell integrates the components into a single standard cell.
`-icgOnly`	Clones only integrated clock gating (ICG) cells along the clock path. ICG cells are standard cells that serve as clock gating cells to allow the switching on and off of clock tree power at the cells' fanout zone. In its simplest form, an ICG cell consists of a Latch and an AND gate that are integrated into a single standard cell to ensure they are placed closely together. Specify this parameter to eliminate unnecessary cell cloning, and to speed up CTS during cell identification. *Default:* Clones all cells along the clock path
`-ignoreDontTouch`	Instructs `ckCloneGate` to ignore the `DontTouch` attribute. `ckCloneGate` will clone instances whether they are marked `DontTouch` or not. *Default*: If you omit this parameter, `ckCloneGate` does not clone `DontTouch` instances.

`-ignorePreplaced`	Instructs `ckCloneGate` to ignore the `FIXED` attribute. `ckCloneGate` will clone instances whether they are fixed or not. *Default*: If you omit this parameter, `ckCloneGate` does not clone `FIXED` instances.
`-ilm`	Does not clone clock gated cells that are inside ILM blocks, or whose sink pins are completely inside ILM blocks. For clock gated cells with a few sink pins inside ILM blocks, and the majority outside ILM blocks, the software performs cloning so that the original clock gated cell drives the sin kpins inside the ILM block, and cloned cells drive the sinkpins outside the ILM block
`-msvAware`	Clones clock gating cells if the power domain of the clock gating cells is different from the power domain of the flops. When specified, CTS clones the cell, and pushes it down to the power domain of the flops. CTS can clone the cell if a level shifter exists, as long as it is a 1 input-1 output level shifter. If you specify this parameter with the `-timingDriven` parameter, CTS takes timing into account when cloning clock gated cells.
`-noUpsizeCell`	Prevents `ckCloneGate` from upsizing cells before cloning. *Default*: If you omit this parameter, `ckCloneGate` upsizes cells before cloning.
`-specOnly`	Restricts cloning to cells specified in the clock tree specification file. *Default*: If you omit this parameter, `ckCloneGate` clones all gating cells.

`-timingDriven`

Takes timing into account when cloning clock gated cells, to determine the optimal level for the clock gating cells.

When specified, the software first determines whether cloning clock gated cells will reduce clock gating violations, before committing any changes. In this way, CTS can place clock gate cells at a higher level to save power, but still meet timing requirements.

Note: The software runs virtual buffering before performing the cloning evaluation and implementation. These buffers will not be present in the netlist after cloning is completed.

If you specify the `-timingDriven` parameter when you are in multi-mode multi-corner analysis mode, CTS clones clock gated cells in the default analysis view only.

If you specify `-timingDriven` with the `-ilm` parameter, the software performs timing-driven clock gate cloning in ILM flows. If you specify `-timingDriven` with `-msvAware`, the software performs timing-driven clock gate cloning in MSV flows.

Note: When you specify `-timingDriven` , the following parameters are not supported:

- `-specOnly`
- `-check`
- `-handleInstGrp`
- `-separateGateFF`
- `-cloneBuf`
- `-handleGatedSinks`
- `-hTreeAware`
- `-enableStrictCloning`

Examples

- The following commands illustrate how to suppress HTML reporting for the specification file `cts.spec` when making a `ckSynthesis` run that involves cloning:

```
setCTSMode -reportHTML false
```

```
specifyClockTree -clkfile cts.spec
ckDeCloneGate
ckCloneGate
ckSynthesis
```

- The following text illustrates the contents of the `ckCloneGate` checking report:

```
Clock: clk
MasterGateInst: 1
Inst : clk_latch, Cell : TLATNCAX2
Nr. Of Sinks: 807
Nr. Of Gates: 0
Estimated Nr. Of Cloning Gates: 116
…
MasterGateInst
+ 116 clk_latch
```

ckDecloneGate

```
ckDecloneGate
     [-clk clockName]
     [-ignorePreplaced]
     [-ignoreDontTouch]
     [-forceReconvergent]
     [-breakLoop]
     [-specOnly]
     [-check]
     [-icgOnly]
     [-handleInstGrp]
     [-file fileName]
     [-detailInfo]
```

Declones (or merges) identical clock gating components with the same inputs, depending on the parameters you specify.

CTS automatically detects integrated clock gating cells and simple combinational gating cells. *Clock gating components* consist of

- Buffers
- Inverters
- AND gates
- OR gates
- Any other logical element (defined in the library) that
  - Appears in the clock tree before clock tree synthesis inserts any buffers or inverters and
  - Drives at least one synchronous pin

The decloning functionality *does not*

- Consider the gating check timing
- Upsize clock gating cells.

You can use this command after loading a clock tree specification file, and before running `ckCloneGate`. For specification file information, see Creating a Clock Tree Specification File in the “Synthesizing Clock Trees” chapter of the *Encounter User Guide*.

Parameters

`-breakLoop`	Instructs CTS to automatically detect loops in clock trees. If CTS detects a loop in a clock tree, CTS ■ Displays a message ■ Breaks the loop ■ Continues to trace through the clock tree
`-check`	Creates a report showing how CTS will perform decloning. The report provides the following information: ■ Clock name ■ Number of equivalent gate instances ■ Instance name ■ Cell name ■ Number of sinks ■ Number of clock gating cells ■ Number of cloned cells to be added to the tree ■ A script that you can add to the clock tree specification file **Note:** For combinational clock gating cells, `-check` reports only the AND/OR gates (not latches).
`-clk` *`clockName`*	Specifies the clock root name defined in the clock tree specification file.a *Default*: If you omit this parameter, `ckDeCloneGate` uses the names of all the clocks in the specification file.
`-detailInfo`	Prints all decloned instances in the log file. If this parameter is not used, then only the summary of decloned instances without instance names is printed.
`-file` *`fileName`*	Saves the information generated by `-check` to the specified file.
`-forceReconvergent`	Forces CTS to trace clock reconvergent paths. If your design contains clock reconvergent paths and you do not specify this parameter, CTS halts and issues an error message.

`-handleInstGrp`	Declones groups of discrete components (such as Latches and AND gates) that are used as clock gating cells. This parameter is useful for designs that do not use integrated clock gating (ICG) cells. The difference between a component group and an ICG cell is that the ICG cell integrates the components into a single standard cell.
`-icgOnly`	Declones only integrated clock gating cells along the clock path. ICG cells are standard cells that serve as clock gating cells to allow the switching on and off of clock tree power at the cells' fanout zone. In its simplest form, an ICG cell consists of a Latch and an AND gate that are integrated into a single standard cell to ensure they are placed closely together. Specify this parameter to eliminate unnecessary cell decloning, and to speed up CTS during cell identification. *Default:* Declones all cells along the clock path
`-ignoreDontTouch`	Instructs `ckDeCloneGate` to ignore the `DontTouch` attribute. `ckDeCloneGate` will clone instances whether they are marked `DontTouch` or not. *Default*: If you omit this parameter, `ckDeCloneGate` does not clone `DontTouch` instances.
`-ignorePreplaced`	Instructs `ckCloneGate` to ignore the `FIXED` attribute. `ckCloneGate` will clone instances whether they are fixed or not. *Default*: If you omit this parameter, `ckCloneGate` does not declone `FIXED` instances.
`-specOnly`	Restricts decloning to cells specified in the clock tree specification file. *Default*: If you omit this parameter, `ckDeCloneGate` declones all gating cells.

Examples

- The following commands illustrate how to suppress HTML reporting for the specification file `cts.spec` when making a `ckSynthesis` run that involves cloning:

```
setCTSMode -reportHTML false
specifyClockTree -clkfile cts.spec
ckDecloneGate
```

```
ckCloneGate
ckSynthesis
```

- The following text illustrates the contents of the `ckDecloneGate` checking report for a single clock gating cell:

```
Clock: clk
EquivGateInst: 1
Inst : G1, Cell : TLATNCAX20
Nr. Of Sinks: 10
Nr. Of Gates: 0
Inst : G2, Cell : TLATNCAX20

EquivGateInst
+ G1
+ G2
…
```

- The following text illustrates the contents of the `ckDecloneGate` checking report for clock gating cells in a group:

```
Clock: clk
EquivGateInst: 1
Inst : clk_and1, Pin: A, Cell : AND2X2
Nr. Of Sinks: 100
Nr. Of Gates: 0
Associated Group Instances: clk_neg1
Inst :clk_and2, Pin : A, Cell : AND2X4
Nr. of Sinks: 95
Nr. Of Gates: 0
Associated Group Instances: clk_neg2

EquivGateInst
+ clk_and1 clk_neg1
+ clk_and2 clk_neg2
```

ckECO

```
ckECO
     [-area]
     [-clk clkName]
     [-report reportName]
     [-preRoute | -clkRouteOnly | -postRoute]
     [-useSpecFileCellsOnly]
     [-dontFixAddedBuffers]
     [-localSkew [-num number]]
     [-fixDRVOnly]
     [-spreadTriggerEdgeDelay]
```

Controls post-CTS optimization.

The `ckECO` command also allows you to reload the buffer or inverter tree from a previous CTS run, if you have exited the Encounter software before routing clock nets and resizing clock tree buffers or inverters. You can also use this command for designs with routed clock and signal wires. The `ckECO` command recognizes RC extraction data imported into the Encounter software with the `spefIn` command.

“Clock gating components” consist of buffers, inverters, AND gates, OR gates, and any other logical element (defined in the library) that appears in the clock tree before CTS synthesis inserts any buffers or inverters.

You can use the `ckECO` command after specifying the clock tree specification file.

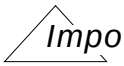
Note: Multi-corner CTS does not support `ckECO -localSkew`.

The following table shows how the `ckECO` command behaves with various `RouteClkNet` and `PostOpt` settings in the clock tree specification file. An X indicates that CTS performs the particular action:

RouteClkNet	PostOpt	CTS Action		
		Route clock nets	Resize buffers and inverters, and refine placement	Correct wires
No (default)	No			
No (default)	Yes (default)		X	X
Yes	Yes (default)	X	X	X
Yes	No	X		

For more information on using the `ckECO` command, see Understanding Post-CTS Clock Tree Optimization, in the “Synthesizing Clock Trees” chapter of the *Encounter User Guide*.

Parameters

`-area`	Reduces the clock tree area only. This parameter cannot be used with `-localSkew`, `-spreadTriggerEdgeDelay`, and `-fixDRVOnly`.
`-clk` *`clkName`*	Specifies the clock root name defined in the clock tree specification file. If you omit a clock name, all clocks in the design will be affected by the `ckECO` command.
`-clkRouteOnly`	Creates clock tree timing analysis results that are based only on clock tree wires, even if other wires (such as signal net wires) exist in the design. *Default*: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.  *Important* If you use this parameter, CTS does not route clock nets, even if `RouteClkNet YES` appears in the clock tree specification file.
`-dontFixAddedBuffers`	Prevents `ckECO` from marking inserted components in the clock tree as `FIXED` after CTS. *Default*: If you omit this parameter, `ckECO` marks all the inserted components of the clock tree as `FIXED`. These components are not moved during IPO.

`-fixDRVOnly`

Attempts to correct maximum transition violations by inserting and resizing buffers, but does not minimize skew.

When in multi-corner mode (`setCTSMode -multiCorner true`), CTS attempts to correct the maximum transition violations on the default active view.

In order to use this parameter, you must add the following statement to your clock tree specification file:

`ForceMaxTran YES`

The default maximum number of buffers that can be inserted is 50. To increase the limit, change the value of the `optAddBufferLimit` statement in the clock specification file.

Default: Does not correct violations (minimizes skew only)

`-localSkew`

Instructs CTS to minimize local skew for the design. (Local skew is the skew between the clock inputs of neighboring flip-flops.)

CTS improves local skew by

- Building a timing graph based on the Encounter timing constraints
- Finding adjacent flip-flop pairs based on all constrained timing endpoints
- Calculating the skew for the clock pins of each adjacent flip-flop pair. The data for the flip-flop pair with the highest skew appears in *`top_level_cell`*`.ctsrpt`. CTS attempts to minimize local skew based upon this data.

Default: If you do not specify this parameter, CTS does not minimize local skew.

Note: You must specify an Encounter timing constraint file, otherwise CTS cannot build a timing graph. The clock you are trying to optimize must be defined in a `create_clock` statement in the Encounter timing constraint file.

Note: When you specify this parameter, CTS improves local skew, though global skew could become greater or smaller—which also affects insertion delay. Also, CPU time increases when you use this parameter, due to the effort required to build the timing graph and identify flip-flop pairs.

`-num` *`number`*

Sets the number of skew values that CTS reports in the skew distribution section of *`top_level_cell`*`.ctsrpt`. Specify `1` or a larger number.

Default: If you do not specify this parameter, only one skew value—the largest—appears in the `.ctsrpt` file.

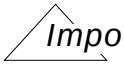
`-postRoute`

Creates clock tree timing analysis results that are based on clock tree wires and signal wires.

Default: If you omit this parameter, CTS considers only clock wires during timing analysis. If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.

Important

If you omit this parameter when running `ckECO` on a routed design, CTS yields less accurate timing results.



Important

If you use this parameter, CTS does not route clock nets, even if `RouteClkNet YES` appears in the clock tree specification file.

Note: CTS halts and displays a message if the design has not been routed.

`-preRoute`

Creates clock tree timing analysis results that are based on a Steiner estimation of the clock tree.

Default: This parameter represents the default behavior of CTS if you do not specify `-preRoute`, `-postRoute`, or `-clkRouteOnly`.

`-report` *`reportName`*

Specifies the name of the clock report file.

Default: If you do not provide a report name, CTS creates a report *`top_level_cell`*`.ctsrpt`, where *`top_level_cell`* is the name of the topmost cell in the design.

`-spreadTriggerEdgeDelay`

Spreads the triggering edge arrival time at the clock pins for all of the flip-flops, to reduce electromagnetic interference (EMI), peak power. While trying to spread the skew, the Encounter software tries to maintain the maximum skew within the specified skew limit.

`-useSpecFileCellsOnly`

Controls where the `ckECO` command looks for candidate inverter or buffer components during resizing with `PostOpt YES` set in the clock tree specification file.

If you specify the `ckECO` command *without* the `-useSpecFileCellsOnly` parameter, CTS looks *only* in the library for candidate cells (that have the same cell footprint) to use when resizing buffers, inverters, or other clock gating components. In this situation, CTS does *not* use the buffers or inverters that appear in the clock tree specification file.

If you specify the `ckECO` command *with* the `-useSpecFileCellsOnly` parameter, CTS looks *only* in the clock tree specification file for buffers and inverters, and *only* in the library for other clock gating components.

The following table describes CTS behavior with the `-useSpecFilesOnly` parameter:

Components that CTS attempts to resize	**Locations that ckECO checks for candidates to replace during resizing with PostOpt YES**	
	Without -useSpecFileCellsOnly	**With -useSpecFileCellsOnly**
Inverter or buffer	Library	Specification file
Clock gating components	Library	Library

Examples

- The following command performs post-routing optimization on clock tree `MCK_GE` and reports the results in the report `MCK_GE.ctsrpt`:

```
ckECO -clk MCK_GE -postRoute -report MCK_GE.ctsrpt
```

ckSynthesis

```
ckSynthesis
     [-clk clkName]
     [-report reportName]
     [-rguide fileName]
     [-macromodel fileName]
     [-check]
     [-forceReconvergent]
     [-dontFixAddedBuffers]
     [-breakLoop | -ignoreLoopDetect]
     [-addOriginalNet]
```

Builds clock trees, routes clock nets, and resizes instances, depending on the parameters you specify. You must create a clock tree specification file before using the `ckSynthesis` command and generating clock reports, clock tree route guides, or macro models for a partition. You can group clocks to balance several independent clocks by using the `ClkGroup` statement in the clock tree specification file.

The `ckSynthesis` command marks inserted components as `FIXED` unless you specify the `-dontFixAddedBuffers` parameter.

“Clock gating components” consist of buffers, inverters, AND gates, OR gates, and any other logical element (defined in the library) that appears in the clock tree before CTS synthesis inserts any buffers or inverters.

When creating a clock tree, the `ckSynthesis` command uses *only* the buffers or inverters listed in the clock tree specification file. If the clock tree specification file contains a `PostOpt YES` statement, the `ckSynthesis` command resizes the components after the clock tree is built. The `ckSynthesis` command chooses candidates to replace the components from

- The clock tree specification file if the object to be resized is a *buffer*
- The library if the object to be resized is a *gate*

Components that CTS attempts to synthesize	Locations that ckSynthesis checks for components with PostOpt YES
Inverters or buffers	Specification file
Clock gating components	Library

Generally, you use this command on a design that has not been routed, and therefore has no signal wires—only clock wires.

Note: CTS will not attempt to resize any gating components—buffers, inverters, flip-flops, or gating cells—that are marked “don’t touch” in the library file.

Parameters

`-addOriginalNet`	Adds the DEF `+ ORIGINAL` property to nets CTS has added to the design.
`-breakLoop`	Instructs CTS to automatically detect loops in clock trees and to exclude pins as necessary. If CTS detects a loop in a clock tree, it displays a message, breaks the loop, and then continues to trace through the clock tree. This parameter disables the `-ignoreLoopDetect` parameter. *Default*: If you do not specify this parameter, CTS checks for loops in the clock tree. If CTS finds loops, it displays a message and stops running.
`-check`	Traces the clock net and reports the clock structure. The report file is *`top_level_cell`*`.cts_trace`. This check is recommended before running CTS. (The Pre-CTS Clock Tree Tracer graphically displays the contents of this trace file.)
`-clk` *`clkName`*	Specifies the clock root name defined in the clock tree specification file. *Default:* If you omit this parameter, CTS chooses the name of the clock in the clock tree specification file.
`-dontFixAddedBuffers`	Sets the placement status of buffers, inverters, flip-flops, and gating cells as `PLACED`. *Default*: Marks all buffers, inverters, flip-flops, and gating cells of the clock tree as `FIXED`. These components are not moved during IPO.
`-forceReconvergent`	Forces CTS to synthesize a clock with self-reconvergence or clocks with crossover points. Without this option, CTS halts and issues errors. To synthesize clocks with crossover points, list such clocks together in the clock tree specification file. To see the results of this parameter in the CTS trace file, use the following clock tree specification file statement: `DetailReport YES`

`-ignoreLoopDetect`	Disables loop detection in the clock tree. When you specify this parameter, CTS runs more quickly, but clock-tree tracing will fail if CTS detects any loops in the clock tree. *Default*: If you do not specify this parameter, CTS checks for loops in the clock tree. If CTS finds loops, it displays a message and stops running.
`-macromodel` *`fileName`*	Specifies the name of the partition macro model instance. This output file is used when creating the top-level partition's clock tree specification file. The macro model file contains the macro model for a partition or a module. A generated macro model helps balance the clocks when you run CTS on the top-level partition. **Note:** When in multi-corner mode (`setCTSMode -multiCorner true`), CTS generate a file that contains macro models for each active analysis view. The last field of the statement indicates the analysis view name, as set by the `set_analysis_view` command.
`-report` *`reportName`*	Specifies the name of the clock report file. *Default*: If you do not provide a clock report name, CTS creates a report *`top_level_cell`*`.ctsrpt`, where *`top_level_cell`* is the design's topmost cell.
`-rguide` *`fileName`*	Specifies the name of the routing guide file. (The routing guide file assigns clock tree preroutes during trial routing.)

Examples

- The following command runs CTS on all clocks and outputs a report file and a routing guide file for use as preroutes or wires during trial routing:

```
ckSynthesis -report clock_out2.report -rguide clock_out2.guide
```

cleanupSpecifyClockTree

`cleanupSpecifyClockTree`

Removes clock tree information that was loaded when you specified the clock tree specification file.

You can use the `cleanupSpecifyClockTree` command after specifying a clock tree specification file, particularly if you want to use a clock tree specification file that you have modified (for example, by removing or adding clock information).

clearClockDisplay

```
clearClockDisplay
```

Removes the graphically displayed clock tree information.

You can use the `clearClockDisplay` command after displaying the clock tree synthesis results.

clockDesign

```
clockDesign
     [-specFile file1 file2 ...]
     [-specViewList {{spec1 view1 view2...} {spec2 view1 view2...}...}]
     [-genSpecOnly fileName]
     [-outDir dirName]
     [-clk clkName]
     [-check]
     [-macromodel fileName]
     [-noDeleteClockTree]
     [-postCTSsdcFile]
     [-incrPostCTSsdcFile]
     [-skipTimeDesign]
```

Automates the CTS portion of the Encounter timing closure flow. The `clockDesign` command streamlines the settings of recommended command modes and parameters. It automatically runs clock tree synthesis based on the default settings of setCTSMode and SDC constraints, and generates the standard clock skew and timing reports.

The `clockDesign` command supports useful skew (`setOptMode -usefulSkew true`). It automatically deletes existing clock trees unless you specify `-noDeleteClockTree`.

If the software is in multi-mode multi-corner (MMMC) analysis mode, and the `setCTSMode -specMultiMode` parameter is set to `true`, only the following `clockDesign` parameters can be used: `-genSpecOnly`, `-outDir`, and `-specViewList`.

You can use the `clockDesign` command after specifying the clock tree specification file.

Important

The `clockDesign` command will not run without a list of buffers or inverters. Specify this information in one of two ways:

- Through the Encounter GUI (the *CTS Cell List* option on the *Design – Design Import – Advanced – IPO/CTS* tab).
- With the following commands:

  `freeCTSCellList`: Clears the previous cell list settings.

  `printCTSCellList`: Displays the current cell list in the Encounter console.

  Note: The above commands are of use only with the `clockDesign` command. Use these commands immediately before running the `clockDesign` command *only* if a cell list has not already been defined in the *Design Import* form.

If you want to see which CTS commands (and parameters) that the `clockDesign` command has used during a particular Encounter session, see the Encounter log file for that session.

You can control the behavior of `clockDesign` and its relationship to the `setCTSMode` command in the following ways:

- If you run `clockDesign` without any parameters, the `clockDesign` command synthesizes the clock trees using the current settings of the `setCTSMode` command and the SDC constraints.
- By changing the default settings of the `setCTSMode` command, the corresponding parameters in the generated clock tree specification file are altered.
- The `setCTSMode` command has certain default settings. You can change the default settings as required for your needs. These settings, however, are global and CTS applies the settings to all the clocks in the design.
- The various settings in the clock tree specification file are commented out by default. You can uncomment and change the settings for specific clocks if required. You can then reload the updated clock tree specification file with the `clockDesign -specFile` parameter and run CTS. (The settings are commented out to give you the flexibility to be able to control various parameters for the remaining clocks that have the global settings defined by `setCTSMode`.)
- The settings in the clock tree specification file take priority over settings defined by `setCTSMode`. The `setCTSMode` command cannot override parameters that are defined in the clock tree specification file.

Parameters

`-check`
: Traces the clock net and reports the clock structure. The report file is *`top_cell_name`*`.cts.trace`.

  Note: This check is recommended before you run gated CTS.

`-clk` *`clkName`*
: Builds clock tree only for the specified clock.

`-genSpecOnly` *`fileName`*
: Stops after generating a clock tree specification file based on the settings of `setCTSMode` and the SDC constraints.

`-incrPostCTSsdcFile` *`fileName`*

Loads the specified incremental SDC file.

Use this parameter to load an incremental SDC file after clock tree synthesis in order to make minor changes, such as changing the `clock_uncertainty` value, or to add the `set_clock_latency` statement.

Note: You can load an incremental SDC file and a post-CTS SDC file (`-postCTSsdcFile`) together. If you do, CTS clears the original SDC file from memory, loads the new post-CTS file, then loads the incremental file.

`-macroModel` *`fileName`*

Specifies the name of the macro model report. Can report after running CTS or after restoring the design and specifying the clock tree specification file.

`-noDeleteClockTree` Specifies that CTS should not remove existing clock trees before running the `ckSynthesis` command.

Default: If you omit this parameter, CTS removes existing clock trees before `ckSynthesis`.

`-outDir` *`dirName`* Creates a directory for the timing and skew reports after the completion of CTS.

`-postCTSsdcFile` *`fileName`*

Loads the specified post-CTS SDC file after clock tree synthesis, before running `timeDesign`.

If you specify this parameter, CTS clears the original SDC file from memory and loads the new SDC file before running `timeDesign -postCTS`.

Note: You can load an incremental SDC file (`-incrPostCTSsdcFile`) and a post-CTS SDC file together. If you do, CTS clears the original SDC file from memory, loads the new post-CTS file, then loads the incremental file.

`-specFile` *`file1 file2 ...`*

Specifies the user-defined clock tree specification file to use to drive CTS. You can specify one or more specification files.

`-specViewList {{`*`spec1 view1 view2...`*`} {`*`spec2 view1 view2...`*`}...}`

Loads the specified clock tree specification files, and uses them to run clock tree synthesis on the specified analysis views.

For example, if you specify:

```
clockDesign -specViewList {{func_mode.spec /
  func_max_view1 func_max_view2} {test_mode.spec /
  func_min_view1 test_min_view1}}
```

The software performs the following actions:

1. Runs CTS using the `func_mode.spec` spec file for the delay corner objects in the `func_max_view1` and `func_max_view2` analysis views.
2. Marks the nets as `dontTouch`.
3. Runs CTS using the `test_mode.spec` spec file for the delay corner objects in the `func_min_view1` and `test_min_view2` analysis views.

Note: The `-specViewList parameter` can be used only with the `-outDir` parameter.

`-skipTimeDesign`

Eliminates the run of `timeDesign -postCTS` in `clockDesign`.

By default, `timeDesign -postCTS` runs within `clockDesign`.

Related Topics

- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*

clockSpiceOut

```
clockSpiceOut
     -clk clockName
     -cdb {cdbFile … | cdbDirectory}
     [-spef fileName]
     -output fileName
```

Creates a SPICE netlist file (containing RC parasitics) that you can use with circuit simulation software to validate

- Skew values and insertion delay for a given clock tree
- Transition times at the inputs of clock buffers for a given clock tree

You can use the SPICE output file

- After routing, when clock tree synthesis, placement, routing, and extraction are completed for your design.
- Before routing, when the clock tree is synthesized and the design is placed.

The `clockSpiceOut` command also generates a name mapping file that maps the internal circuit names generated by the SPICE output commands to the corresponding node names in the SPEF file. This mapping file name consists of the SPICE netlist filename and the filename extension `.nm`. This file is created in the directory in which you started the Encounter™ software.

Clock Waveform Derivation

Clock waveforms are derived from the SDC file. If SDC data are not available, `clockSpiceOut` uses the following values:

- Rising edge: 0 ns
- Falling edge: 12 ns
- Rise transition: 2 ns
- Fall transition: 2 ns
- Clock period: 20 ns

SPEF File Generation

There are several ways to generate the SPEF file to be used with `clockSpiceOut`:

- Route the design with `trialRoute` and use `rcOut -spef` *`fileName`* to create the SPEF file.
- Route the design with the NanoRoute™ Ultra SoC routing solution and use QRC extraction software to create the SPEF file. (This is the most accurate method.)
- Use the `clockSpiceOut` command without the `-spef` parameter. CTS uses the `rcOut` command to create the SPEF file.

Note: CTS cannot produce SPICE output files for

- Multiple clock trees
- Hierarchical clock trees
- Crossover clocks

Parameters

`-cdb {`*`cdbFile`* … | *`cdbDirectory`*`}`

Specifies cdb files or a directory containing cdb file(s). If you specify a directory name, CTS looks for all files that end with `.cdb`, `.cdB`, or `.udn` filename extensions.

The cdb file or directory provide subcircuit netlists for the logical and sequential elements in the clock tree (such as clock buffers, logical gates, and registers).

Important

The cdb files you specify *must* be in decrypted/text format.The `clockSpiceOut` command does not support cdb files in binary/encrypted format.

`-clk` *`clockName`*

Specifies the name of the root clock's net name, the root clock pin name, or the root clock instance pin name.

`-output` *`fileName`*

Specifies the name of the output SPICE netlist file.

`-spef` *`fileName`*

Specifies the SPEF file containing the RC parasitics of the clock tree.

If you omit this parameter, CTS uses the Encounter `rcOut` command to generate the SPEF file.

Note: If you have not extracted the parasitics for your design, CTS displays an error message.

Examples

- The following example shows the text of a SPICE netlist file created by `clockSpiceOut`:

```
*
*  Dump Spice.  For clk
*
V1 VDD 0   2.2000e+00
V2 GND 0   0.0000e+00
V3 n1:0 0 PULSE (   0.0000e+00   2.2000e+00   0.0000e+00   1.0000e-09   1.0000
e-09   4.0000e-09   2.0000e+01 )
V4 VLwave 0   0.0000e+00
.tran    1.0000e-10   1.0000e-07
R1 n1:1 n1:0   8.2813e-01
R2 n1:2 n1:1   4.0000e+00
R3 n1:3 n1:2   1.3250e+00
...
C1 n1:0 0   3.4000e-17
C2 n1:1 0   3.4000e-17
C3 n1:2 0   6.4500e-17
...
* INSTANCE: clk__L1_I0
M1 n1:14 n1:15 VDD VDD bdaac_cadmos_pmos  L=   4.0000e-07  W=   3.0000e-06  AD
=   6.0000e-12  AS=   6.0000e-12  PD=   1.0000e-05 PS=   1.0000e-05 NRD=   3.3
333e-01 NRS=   3.3333e-01 $x1=  2.1900e+02 $y1=  2.6500e+02 $x2=  2.2700e+02 $
y2=  2.6600e+02

M2 n1:14 n1:15 GND GND bdaac_cadmos_nmos  L=   4.0000e-07  W=   1.0000e-06  AD
=   2.0000e-12  AS=   2.0000e-12  PD=   6.0000e-06 PS=   6.0000e-06 NRD=   1.0
000e+00 NRS=   1.0000e+00 $x1=  2.1900e+02 $y1=  2.6500e+02 $x2=  2.2700e+02 $
y2=  2.6600e+02

...

.option
.option
.MEAS TRAN n1:0_n1:53_delay TRIG V(n1:0) VAL = 0.5*VDD TD = 0n RISE = 1 TARG V
(n1:53) VAL = 0.5*VDD RISE = 1
.MEAS TRAN n1:125_rise TRIG V(n1:125) VAL = 0.1*VDD TD = 0n RISE = 1 TARG V(n1
:125) VAL = 0.9*VDD RISE = 1
...
```

```
.end
```

- The following example shows the text of a mapping file created by `clockSpiceOut`:

```
n1:0  clk
n1:1  clk:2
n1:2  clk:0
n1:3  clk:1

...

n1:12  clk__L1_I0:A
n1:13  clk__L1_I0:Minv2.mp
n1:14  clk__L1_I0:Z
n1:15  clk__L1_I0:mid
n1:16  clk__L1_I0:Minv2.mn

...
```

createClockTreeSpec

```
createClockTreeSpec
     -file clkSpecFile
     [-ignoreDontUse]
     [-bufFootprint bufFootprintName]
     [-invFootprint invFootprintName]
     [-bufferList bufferName… inverterName…]
     [-routeClkNet]
     [-views {view1 view2 ...}]
```

Extracts and translates clock-related timing constraint information from the Encounter timing constraints file and adds that data to the clock tree specification file. The `createClockTreeSpec` command also identifies through-pins for divided clocks automatically.

For each clock constraint defined in the Encounter timing constraints file, CTS creates a separate clock root section in the clock tree specification file.

The `createClockTreeSpec` command also

- Groups clocks in the same clock domain, as necessary.
- Analyzes the SDC constraint file, automatically translates certain constraints, and adds those constraints to the clock tree specification file.

  The following table shows the SDC constraints that the `createClockTreeSpec` command automatically translates into clock tree specification file statements. The table also shows the default values used in those statements if an SDC constraint does not exist:

SDC Constraint	Clock Tree Specification File Statement (default)
`create_clock`	`AutoCTSRootPin`
`set_clock_transition`	`SinkLeafTran and BufMaxTran` (Default: 400 ps)
`set_clock_latency` *`value`*	`MaxDelay` (Default: clock period) `MinDelay` (Default: 0)
`set_clock_latency -source` *`value`*	`SrcLatency` *`value`* `ns`
`set_clock_uncertainty`	`MaxSkew` (Default: 300 ps)
`create_generated_clock`	Add `ThroughPin` when necessary

The various settings in the clock tree specification file are commented out by default. You can uncomment and change the settings for specific clocks if required. (The settings are commented out to give you the flexibility to be able to control various parameters for the remaining clocks that have the global settings defined by `setCTSMode`.) By changing the default settings of the `setCTSMode` command, the corresponding parameters in the generated clock tree specification file are altered.

For each clock tree specification file, you can specify one of the following sets of buffer or inverter footprint information:

- One buffer footprint name
- One inverter footprint name
- One buffer footprint name and one inverter footprint name

You can use the `createClockTreeSpec` command after extracting RC data for the design, and after you are confident that the timing constraints are appropriate. Timing constraints must be loaded into the design before using this command. The design does not need to be placed.

Parameters

`-bufferList` *`bufferName`*… *`inverterName`*…

Allows you to specify a list of buffer and inverter names if your timing library does not have footprint information. For example:

```
createClockSpec -bufferList BUFX1 BUFXL INVX8
```

Note: This parameter is *not* affected by `-ignoreDontUse`.

`-bufFootprint` *`bufFootprintName`*

Allows you to specify buffer cell name(s) or buffer footprint name(s).

Cells in the library marked as “don’t use” are not added to the buffer list in the clock tree specification file unless you specify the `-ignoreDontUse` parameter.

`-file` *`clkSpecFile`* Specifies the name of the clock tree specification file.

`-ignoreDontUse` Instructs CTS to ignore the “don’t use” attribute for buffers and inverters in the `.lib` file.

Note: This parameter applies *only* to `-bufFootprint` and `-invFootprint`.

`-invFootprint` *`invFootprintName`*

Allows you to specify inverter cell name(s) or inverter footprint name(s).

Cells in the library marked as “don’t use” are not added to the buffer list in the clock tree specification file unless you specify the `-ignoreDontUse` parameter.

`-routeClkNet` Adds the `RouteClkNet YES` statement to a clock tree specification file.

`-views` *`view1 view2 ...`*

Generates a single specification file that is based on the specified analysis views.

Examples

- The following commands show how to use the `createClockTreeSpec` command in combination with other CTS commands:

```
loadTimingCon sdc_timing
createClockTreeSpec -output Mychip.ctstch -bufFootprint buf -invFootprint inv
specifyClockTree -clkfile Mychip.ctstch
ckSynthesis -report Mychip.ctsrpt
saveClockNets -output Mychip.ctsntf
```

deleteClockTree

```
deleteClockTree
     {-clk clkName | -all}
     [-deleteFEOnly]
     [-file fileName]
```

Removes the synthesized clock tree and the inserted clock buffers and clock nets.

Note: The following exceptions apply to the behavior of the `deleteClockTree` command:

- Unlike other clock tree generation tools, CTS does *not* remove preexisting clock trees from a design. Therefore, if you want to remove previously created clock trees, you must use the `deleteClockTree` command.
- The `deleteClockTree` command does not remove `FIXED` buffers and inverters from a design. To remove them, specify the following command to change their status from `FIXED` to `PLACED`, before using the `deleteClockTree` command:

  ```
  changeClockStatus -all -noFixedBuufers
  ```

- When CTS builds a clock tree in a hierarchical design, CTS adds one or more ports to the hierarchical design. The `deleteClockTree` command does not remove such ports. See `setCTSMode -honorFence` for more information.

You can use the `deleteClockTree` command after performing clock tree synthesis on the design.

Parameters

`-all`	Deletes the synthesized clock tree and any inserted clock buffers and clock nets for all clocks in the design.
`-clk clkName`	Specifies the clock root name in the clock tree specification file.
`-deleteFEOnly`	Deletes only the buffers that the Encounter software inserted.
`-file fileName`	Specifies the output file that contains the hierarchical instance names that were deleted from the design.

Examples

- The following command removes the synthesized clock buffers and nets from clock tree `MCK_GE`:

  ```
  deleteClockTree -clk MCK_GE
  ```

displayClockMinMaxPaths

```
displayClockMinMaxPaths
     -clk clkName
     [-preRoute | -clkRouteOnly | -postRoute]
     [-max number]
     [-min number]
     [-fall | -rise]
     [-view viewName]
     [-all]
```

Graphically displays the clock paths with the minimum and maximum delays. The clock path with the smallest delay is colored blue, and the clock path with the largest delay is colored red.

You can use the `displayClockMinMaxPaths` command after performing clock tree synthesis on the design.

Parameters

`-all`	Displays all of the clock paths for the specified clock. The software prints the output to the log file sorted by the type of delay (by default, `-rise`) and timing results (by default, `-preRoute`) specified. *Caution* ***CAUTION: Specifying this parameter for a large design creates an extremely large output in the log file, which can take more than 10 minutes to print.***
`-clk clkName`	Specifies the clock root name defined in the clock tree specification file. *Default:* If you omit this parameter, CTS uses the names of all clocks in the clock tree specification file.

`-clkRouteOnly`

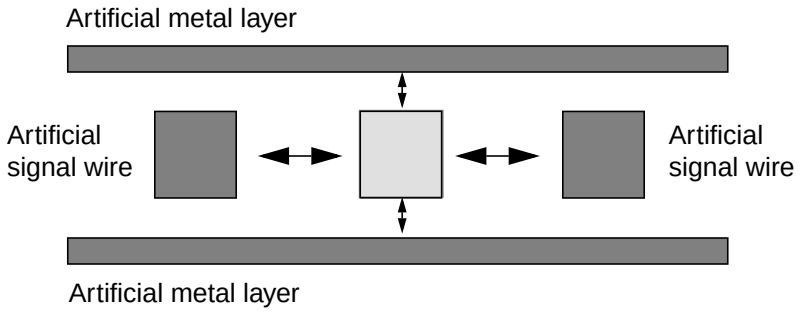
Displays clock tree timing analysis results that are based only on clock tree wires, even if other wires (such as signal net wires) exist in the design.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.

This parameter extracts coupling capacitance for clock wires based on

- Artificial parallel signal wires on either side of the clock wires and
- Artificial top and bottom metal layers of infinite size

In the following figure, the arrows indicate where CTS extracts estimated coupling capacitance based upon the artificial layers and signal wires:



The goal of this method of coupling capacitance extraction is to achieve results as close to those of the `-postRoute` parameter as possible in the absence of actual signal wires.

`-fall | -rise`

Determines the type of delay that CTS reports for each clock you specify.

Default: If you omit these parameters, CTS behaves as if you had specified `-rise`.

`-max` *`number`*

Defines the number of maximum delays that CTS reports in the Encounter console and in the CTS log file.

Default: If you omit this parameter, CTS reports one maximum delay value.

`-min` *`number`*
: Defines the number of minimum delays that CTS reports in the Encounter console and in the CTS log file.

  Default: If you omit this parameter, CTS reports one minimum delay value.

`-postRoute`
: Displays clock tree timing analysis results that are based on clock tree wires and signal wires.

  Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.

  You should use this parameter if your design is completely routed (that is, all the clock nets and signal nets are routed).

  CTS creates postrouting timing data using

  - Routing data based on information using `extractRC`
  - SPEF data imported with `spefIn`

  Note: CTS halts and displays a message if the design has not been routed.

`-preRoute`
: Displays clock tree timing analysis results that are based on a Steiner estimation of the clock tree.

`-view` *`viewName`*
: Displays the clock paths with the minimum and maximum delays for the specified multi-corner analysis view.

  To use this parameter, CTS must be in multi-corner mode (`setCTSMode -multiCorner true`).

  Default: Displays the clock paths with the minimum and maximum delays for the default analysis view.

Examples

- The following commands show how to display minimum and maximum delays for the path `clk`:

```
ckSynthesis -clk clk
displayClockMinMaxPaths -clk clk
```

- The following example shows information (found in the `encounter.log` file) on paths that correspond to minimum and maximum rise/fall delay:

```
displayClockMinMaxPaths -clk clk

**** Clock clk (Nr . of Sink: 12345) ****

*Minimum Clock Path
StartPoint     : clk
EndPoint      : core/data_reg_1/CK
Delay r/f (ns): 4.01/4.11

Object name               Delta r/f (ns)  Sum r/f (ns) Slew (ns)
-----------------------------------------------------------------------
clk                              0.012/0.012  0.012/0.012
clk_L1_I0 A -> Y       0.301/0.303   0.313/0.315  0.12/0.12
....
core/data_reg_1/CK    0.0/0.0          4.01/4.11

*Maximum Clock Path
StartPoint     : clk
EndPoint      : core/data_reg_111/CK
Delay r/f (ns): 4.11/4.21

Object name                     Delta r/f (ns)  Sum r/f (ns) Slew (ns)
-----------------------------------------------------------------------
--
clk                                    0.012/0.012  0.012/0.012
clk_L1_I0 A -> Y              0.301/0.303   0.313/0.315  0.12/0.12
....
core/data_reg_111/CK       0.0/0.0           4.11/4.21
```

displayClockPhaseDelay

```
displayClockPhaseDelay
     [-clk clkName]
     [-preRoute | -clkRouteOnly | -postRoute]
     [-max delay]
     [-min delay]
     [-sinkonly]
     [-view viewName]
```

Graphically displays the phase delays for clock instances. Colors at the red end of the spectrum indicate the greatest phase delay. Colors at the blue end of the spectrum indicate the smallest phase delay.

You can use the `displayClockPhaseDelay` command after performing clock tree synthesis on the design.

Parameters

`-clk` *`clkName`*	Specifies the clock root name defined in the clock tree specification file. *Default:* If you omit this parameter, CTS displays the names of all clocks in the clock tree specification file.

`-clkRouteOnly`

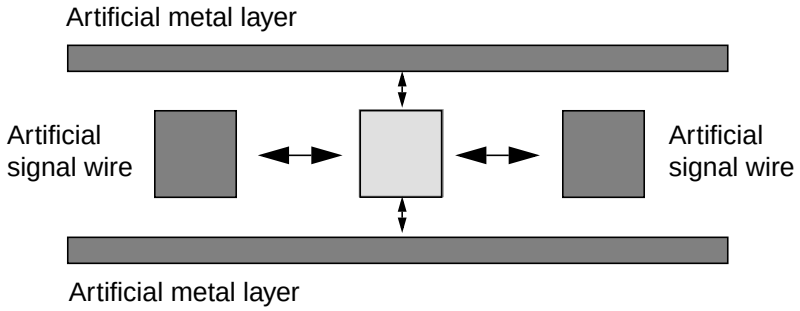
Displays clock tree timing analysis results that are based only on clock tree wires, even if other wires (such as signal net wires) exist in the design.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.

This parameter extracts coupling capacitance for clock wires based on

- Artificial parallel signal wires on either side of clock wires
- Artificial top and bottom metal layers of infinite size

In the following figure, the arrows indicate where CTS extracts estimated coupling capacitance based upon the artificial layers and signal wires:



The goal of this method of coupling capacitance extraction is to achieve results as close to those of the `-postRoute` parameter as possible in the absence of actual signal wires.

`-max` *`delay`*

Sets a maximum delay, in picoseconds. If a buffer's delay is greater than *`delay`*, the path connected to the clock buffer is red.

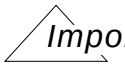
`-min` *`delay`*

Sets a minimum delay, in picoseconds. If a buffer's delay is less than *`delay`*, the path connected to the clock buffer is blue.

`-postRoute`

Displays clock tree timing analysis results that are based on clock tree wires and signal wires.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.



Important

You should use this parameter if your design is completely routed (that is, all the clock nets and signal nets are routed).

CTS creates postrouting timing data using

- Routing data based on information using `extractRC`
- SPEF data imported with `spefIn`

Note: CTS halts and displays a message if the design has not been routed.

`-preRoute`

Displays the clock tree based on timing analysis using a Steiner estimation of the clock tree.

`-sinkonly`

Displays only the flip-flops.

`-view` *`viewName`*

Displays the phase delays for clock instances in the specified multi-corner analysis view.

To use this parameter, CTS must be in multi-corner mode (`setCTSMode -multiCorner true`).

Default: Displays the phase delays for clock instances in the default analysis view.

displayClockTree

```
displayClockTree
     [-clk clkName [-level levelNumber | -allLevel]]
     [-preRoute | -clkRouteOnly | -postRoute]
     [-skew]
     [-view viewName]
```

Graphically displays the results of CTS.

You can use the `displayClockTree` command after performing clock tree synthesis on the design.

Parameters

`-allLevel`	Displays gated and nongated clock trees in the Physical view. You do not need to specify the `-skew` or `-level` parameters with `-allLevel`. **Note:** You cannot use this parameter with the `-level` parameter.
`-clk` *`clkName`*	Specifies the clock root name defined in the clock tree specification file. *Default:* If you omit this parameter, CTS displays the names of all clocks in the clock tree specification file.

`-clkRouteOnly`

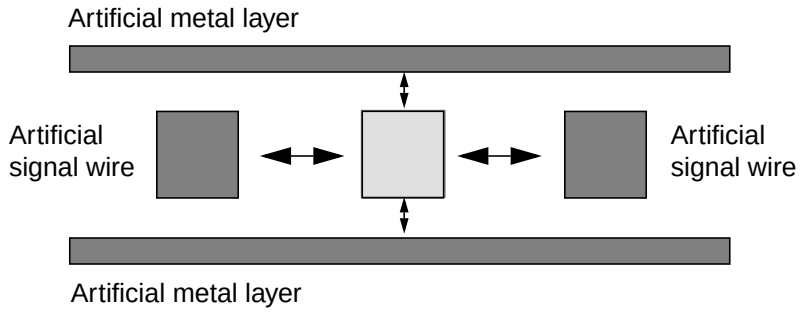
Displays clock tree timing analysis results that are based only on clock tree wires, even if other wires (such as signal net wires) exist in the design.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.

This parameter extracts coupling capacitance for clock wires based on

- Artificial parallel signal wires on either side of the clock wires and
- Artificial top and bottom metal layers of infinite size

In the following figure, the arrows indicate where CTS extracts estimated coupling capacitance based upon the artificial layers and signal wires:



The goal of this method of coupling capacitance extraction is to achieve results as close to those of the `-postRoute` parameter as possible in the absence of actual signal wires.

`-level` *`levelNumber`*

Specifies the clock tree level (expressed as an integer) assigned in the clock tree specification file.

Default: If you omit this parameter, CTS selects the lowest level of the clock tree (highest level number).

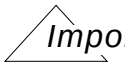
Note: You cannot use this parameter with the `-allLevel` parameter.

Note: You cannot use this parameter with a gated clock tree.

`-postRoute`

Displays clock tree timing analysis results that are based on clock tree wires and signal wires.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.



Important

You should use this parameter if your design is completely routed (that is, all the clock nets and signal nets are routed).

CTS creates postrouting timing data using

- Routing data based on information using `extractRC`
- SPEF data imported with `spefIn`

Note: CTS halts and displays a message if the design has not been routed.

`-preRoute`

Displays the clock tree based on timing analysis using a Steiner estimation of the clock tree.

`-skew`

Displays the clock skew. The clock paths and color-coded skews are displayed.

Colors at the red end of the spectrum indicate the greatest phase delay; colors at the blue end of the spectrum indicate the smallest phase delay.

`-view` *`viewName`*

Displays clock tree results for the specified multi-corner analysis view.

To use this parameter, CTS must be in multi-corner mode (`setCTSMode -multiCorner true`).

Default: Displays clock tree results for the default analysis view.

Examples

- The following command displays the skew results of CTS on all clocks:

```
displayClockTree -skew
```

displayClockTreeMinMaxPaths

```
displayClockTreeMinMaxPaths
     -clk clkName
     -pin pinName
     [-preRoute | -clkRouteOnly | -postRoute]
     [-fall | -rise]
     [-view viewName]
```

Graphically displays the shortest and longest clock tree paths through a specified pin. The clock path with the smallest delay is colored blue, and the clock path with the largest delay is colored red.

You can use the `displayClockTreeMinMaxPaths` command after performing clock tree synthesis on the design.

Parameters

`-clk clkName`	Specifies the clock root name defined in the clock tree specification file. *Default:* If you omit this parameter, CTS uses the names of all clocks in the clock tree specification file.

`-clkRouteOnly`

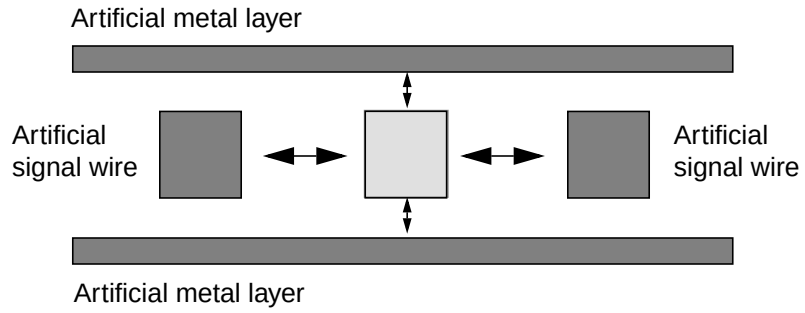
Displays clock tree timing analysis results that are based only on clock tree wires, even if other wires (such as signal net wires) exist in the design.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.

This parameter extracts coupling capacitance for clock wires based on

- Artificial parallel signal wires on either side of the clock wires and
- Artificial top and bottom metal layers of infinite size

In the following figure, the arrows indicate where CTS extracts estimated coupling capacitance based upon the artificial layers and signal wires:



The goal of this method of coupling capacitance extraction is to achieve results as close to those of the `-postRoute` parameter as possible in the absence of actual signal wires.

`-fall | -rise`

Determines the type of delay that CTS reports for each clock you specify.

Default: If you omit these parameters, CTS behaves as if you had specified `-rise`.

`-pin`

CTS reports information for all the paths that pass through the specified pin.

`-postRoute`

Displays clock tree timing analysis results that are based on clock tree wires and signal wires.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.

Important

You should use this parameter if your design is completely routed (that is, all the clock nets and signal nets are routed).

CTS creates postrouting timing data using

- Routing data based on information using `extractRC`
- SPEF data imported with `spefIn`

Note: CTS halts and displays a message if the design has not been routed.

`-preRoute`

Displays clock tree timing analysis results that are based on a Steiner estimation of the clock tree.

`-view` *viewName*

Displays the shortest and longest clock tree paths through a specified pin for the specified multi-corner analysis view.

To use this parameter, CTS must be in multi-corner mode (`setCTSMode -multiCorner true`).

Default: Displays the shortest and longest clock tree paths through a specified pin for the default analysis view.

Examples

- The following example (found in the `encounter.log` file) illustrates how CTS reports information on data that goes through a specific pin:

```
displayClockTreeMinMaxPaths -pin  top/core/ack_reg/CLK

** Clock /top/clk_cntr/clkMux/Y (Nr.of Sink under top/core/ack_reg/CLK:  1) **
*Minimum Clock Path
StartPoint     : top/clk_cntr/clkMux/Y
EndPoint      : top/core/ack_reg/CLK
Delay r/f (ns): 5.16/5.22

Object name                      Delta r/f (ns)  Sum r/f (ns) Slew (ns)
```

```
--------------------------------------------------------------------------------
scan_clk                              0.022/0.032  0.022/0.032
scan_clk_L1_I0 A -> Y       0.301/0.303   0.323/0.335  0.09/0.11
....
top/core/ack_reg/CLK         0.0/0.0          5.16/5.22

*Maximum Clock Path
StartPoint     : top/clk_cntr/clkMux/Y
EndPoint      : top/core/ack_reg/CLK
Delay r/f (ns): 5.16/5.22

Object name                         Delta r/f (ns)  Sum r/f (ns) Slew (ns)
--------------------------------------------------------------------------------
scan_clk                              0.022/0.032  0.022/0.032
scan_clk_L1_I0 A -> Y       0.301/0.303   0.323/0.335  0.09/0.11
....
top/core/ack_reg/CLK         0.0/0.0          5.16/5.22
```

fixClockExcludedNetDRV

`fixClockExcludedNetDRV`

Specifies that CTS must consider the maximum transition constraints on pins listed in the `ExcludedPin` statement in the clock tree specification file.

You can use the `fixClockExcludedNetDRV` command within an existing Encounter session only after using the `ckSynthesis` command.

getCTSMode

```
getCTSMode
     [-addClockRootProp]
     [-fixLeafInst]
     [-fixNonLeafInst]
     [-honorFence]
     [-moveGate]
     [-nameAppendParentInClones]
     [-nameSingleDelim]
     [-opt]
     [-optAddBuffer]
     [-optLatency]
     [-optLatencyMoveGate {true | false}]
     [-powerAware]
     [-quiet]
     [-reportHTML]
     [-routeBottomPreferredLayer]
     [-routeClkNet]
     [-routeGuide]
     [-routeLeafBottomPreferredLayer]
     [-routeLeafNonDefaultRule]
     [-routeLeafPreferredExtraSpace]
     [-routeLeafRouteTypeOnGate]
     [-routeLeafShielding]
     [-routeLeafTopPreferredLayer]
     [-routeNonDefaultRule]
     [-routePreferredExtraSpace]
     [-routeShielding]
     [-routeTopPreferredLayer]
     [-synthLatencyEffort {high | low}]
     [-traceAsyncSRPinAsLeaf {true | false}]
     [-traceBlackBoxPinAsLeaf {true | false}]
     [-traceDPinAsLeaf]
     [-traceHonorConstants]
     [-traceIoPinAsLeaf]
     [-traceTriStateEnablePinAsLeaf {true | false}]
     [-useLefACLimit]
     [-useLibMaxCap]
     [-useLibMaxFanout]
     [-verbose]
```

Displays the following information about a specified clock tree synthesis mode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)

- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the clock tree synthesis mode parameters.

Parameters

parameter_names	Displays information for the specified parameters. You can specify one or more parameters. See setCTSMode for descriptions of the clock tree synthesis parameters you can specify.
-quiet	Displays the current settings for the specified parameters in Tcl list format only. If you specify -quiet without any parameters, the software displays the current settings of all setCTSMode parameters in Tcl list format.

Examples

- The following command displays the current setting for the -honorFence parameter:

  ```
  getCTSMode -honorFence
  ```

  The software displays the following information:

  ```
  -honorFence false                          # bool, default=false
  false
  ```

- The following command displays the current settings for the -routeTopPreferredlayer and -routeBottomPreferredlayer parameters:

  ```
  getCTSMode -routeTopPreferredlayer -routeBottomPreferredlayer
  ```

  The software displays the following information:

  ```
  -routeBottomPreferredLayer 3               # int, default=3
  -routeTopPreferredLayer 4                  # int, default=4

  {routeBottomPreferredLayer 3} {routeTopPreferredLayer 4}
  ```

- The following command displays the current setting for the -honorFence command in Tcl format only:

  ```
  getCTSMode -honorFence -quiet
  ```

  The software displays the following information:

```
false
```

- The following command displays the current settings for the `-routeTopPreferredlayer` and `-routeBottomPreferredlayer` parameters in Tcl format only:

  ```
  getCTSMode -routeTopPreferredlayer -routeBottomPreferredlayer -quiet
  ```

  The software displays the following information:

  ```
  {routeBottomPreferredLayer 3} {routeTopPreferredLayer 4}
  ```

- The following command displays the current settings for all `setCTSMode` parameters in Tcl format only:

  ```
  getCTSMode -quiet
  ```

  The software displays the following information:

  ```
  {addClockRootProp false} {fixLeafInst true} {fixNonLeafInst false} {honorFence
  false} {moveGate true} {multiCorner true} {nameAppendParentInClones false}
  {nameSingleDelim false} {opt true} {optAddBuffer false} {optArea false}
  {optLatency false} {optLatencyMoveGate true} {powerAware false} {reportHTML
  false} {routeBottomPreferredLayer 3} {routeClkNet false} {routeGuide {true}}
  {routeLeafBottomPreferredLayer 3} {routeLeafNonDefaultRule {}}
  {routeLeafPreferredExtraSpace 1} {routeLeafRouteTypeOnGate true}
  {routeLeafShielding {}} {routeLeafTopPreferredLayer 4} {routeNonDefaultRule
  {}} {routePreferredExtraSpace 1} {routeShielding {}} {routeTopPreferredLayer
  4} {routeWithFiller false} {specMultiMode false} {synthLatencyEffort
  low}{traceAsyncSRPinAsLeaf false} {traceDPinAsLeaf false} {traceHonorConstants
  false} {traceIoPinAsLeaf false} {traceOverlapCheck true}
  {traceTriStateEnablePinAsLeaf false} {useLefACLimit false} {useLibMaxCap
  false} {useLibMaxFanout false} {verbose false}
  ```

refineClockTreeCellHalo

```
refineClockTreeCellHalo
```

Refines placement on clock instances so that the cell halo is observed. It excludes the leaf instances.

Examples

- The following command refines placement on clock instances:

  ```
  refineClockTreeCellHalo
  ```

reportClockTree

```
reportClockTree
     [-clk clkName]
     [-preRoute | -clkRouteOnly | -postRoute]
     [-localSkew]
     [-report reportName]
     [-rguide fileName]
     [-macromodel macroFilename]
     [-num number]
     [-view viewName]
```

Reports the results of clock tree synthesis. By default, the report includes the following information about the clock tree:

- Library information, such as the clock name, mode, library name, operating condition and PVT values.
- Clock tree structure information, such as the number of subtrees, sinks, and buffers, and the maximum and minimum trigger edge delay at sinks.
- Delay, skew, and transition values, such as the rise and fall phase delay, skew, buffer transition, and sink transition values.
- Maximum transition time violations
- Skew distribution information, such as the input and output delay ranges.
- Macro model information, such as the maximum and minimum trigger edge delay at sink values.
- AC current density violations

The `reportClockTree` command also reports power information if you specify `setCTSMode -powerAware true` and include the `Period` statement in the clock tree specification file. The report includes information on the total net switching power, internal clock instances power, internal leaf pin power, and leakage power. For more information, see "Power Information" in the "CTS Report Descriptions" section of the "Synthesizing Clock Trees" Chapter of the *Encounter User Guide*.

In multi-corner mode (`setCTSMode -multiCorner true`), the `reportClockTree` command reports the trigger skew value for all active analysis views in the design, and reports other analysis information, such as the phase delay and transition values, only for the default analysis view.

Note: Multi-corner CTS does not support `reportClockTree -localSkew`.

Parameters

`-clk` *`clkName`*

Specifies the clock root name defined in the clock tree specification file.

Default: If you omit this parameter, CTS selects all clocks in the clock tree specification file.

`-clkRouteOnly`

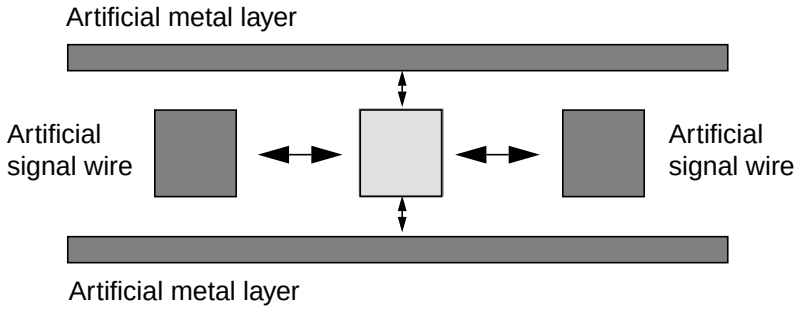
Displays clock tree timing analysis results that are based only on clock tree wires, even if other wires (such as signal net wires) exist in the design.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.

This parameter extracts coupling capacitance for clock wires based on

- Artificial parallel signal wires on either side of the clock wires and
- Artificial top and bottom metal layers of infinite size

In the following figure, the arrows indicate where CTS extracts estimated coupling capacitance based upon the artificial layers and signal wires:



The goal of this method of coupling capacitance extraction is to achieve results as close to those of the `-postRoute` parameter as possible in the absence of actual signal wires.

`-localSkew`

Instructs CTS to report local skew values for the design. (Local skew is the skew between the clock inputs of neighboring flip-flops.)

CTS improves local skew by

- Building a timing graph based on the Encounter timing constraints
- Finding adjacent flip-flop pairs based on all constrained timing endpoints
- Calculating the skew for the clock pins of each adjacent flip-flop pair. The data for the flip-flop pair with the highest skew appears in the *`top_level_cell`*`.ctsrpt`. CTS attempts to minimize local skew based upon this data.

You must specify an Encounter timing constraint file, otherwise CTS cannot build a timing graph. The clock for which you want CTS to report local skew must be defined in a `create_clock` statement in the timing constraint file.

Default: If you do not specify this parameter, CTS does not report local skew.

Note: In multi-corner mode, (`setCTSMode -multiCorner true`, CTS does not support the `-localSkew` parameter.

`-macromodel` *`macroFilename`*

Specifies the name of the macro model report. You can generate a macro model report after running CTS, or after restoring the design and specifying the clock tree specification file.

In multi-corner mode, (`setCTSMode -multiCorner true`, CTS generate a file that contains macro models for each active analysis view. The last field of the statement indicates the analysis view name, as set by the `set_analysis_view` command.

`-num` *`number`*

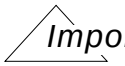
Sets the number of skew values that CTS reports in the local skew report section of the *`top_level_cell`*`.ctsrpt`. Specify `1` or a larger number.

Default: If you do not specify this parameter, only one skew value—the largest—appears in the `.ctsrpt` file.

`-postRoute`

Reports clock tree timing analysis results that are based on clock tree wires and signal wires.

Default: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.



Important

You should use this parameter if your design is completely routed (that is, all the clock nets and signal nets are routed).

CTS creates postrouting timing data using

- Routing data based on information using `extractRC`
- SPEF data imported with `spefIn`

Note: CTS halts and displays a message if the design has not been routed.

`-preRoute`

Reports clock tree timing analysis results that are based on a Steiner estimation of the clock tree.

Default: If you do not specify `-preRoute`, `-postRoute`, or `-clkRouteOnly`, CTS reports results as if you had specified `-preRoute.`

`-report` *`reportName`*

Specifies the name of a clock report file.

`-rguide` *`fileName`*

Specifies the name of the clock routing guide file. This file assigns preroutes during trial routing.

`-view` *`viewName`*

Reports timing analysis information for the specified multi-corner analysis view.

To use this parameter, CTS must be in multi-corner mode (`setCTSMode -multiCorner true`).

Default: Reports the trigger skew for all active analysis views in the design, and reports other analysis information, such as the phase delay and transition values, only for the default analysis view.

Examples

- The following command reports to a file the results of CTS on clock `CLK`:

  ```
  reportClockTree -clk CLK -report clock_CLK.report
  ```

- In mutli-corner mode, the previous command generates a clock timing report that includes the following analysis information for the default analysis view (`view1`), and the trigger skew values for each of the active analysis views in the design:

  ```
  #Analysis View: view1
  **********Clock CLK Pre-Route Timing Analysis***************
  Nr. of Subtrees                 :1
  Nr. of Sinks                    :23200
  Nr. of Buffer                   :542
  Nr. of Level (including gates)  :8
  Max trig. edge delay at sink(R) :FF_34/CK 1347.7(ps)
  min trig. edge delay at sink(R) :FF_12/CK 1210.3(ps)

                              (Actual)             (Required)
  Rise Phase Delay           :1210.3~1347.7(ps)    0~1000(ps)
  Fall Phase Delay           :1219.7~1360.6(ps)    0~1000(ps)
  Trig. Edge Skew            :137.4(ps)            50(ps)
  Rise Skew                  :137.4(ps)
  Fall Skew                  :140.9(ps)
  Max. Rise Buffer Tran.     :500(ps)              300(ps)
  ...
  Min. Rise Sink Tran.       :143.6(ps)            0(ps)
  Min. Fall Sink Tran.       :136.5(ps)            0(ps)

  view view1 : skew = 137.4ps
  view view2 : skew = 132.1ps
  view view3 : skew = 112.5ps
  ```

- In mutli-corner mode, the following command generates a clock timing report for the analysis view `view2`:

  ```
  reportClockTree -view view2
  ```

  The resulting report file contains the following analysis information:

  ```
  #Analysis View: view2
  **********Clock CLK Pre-Route Timing Analysis***************
  Nr. of Subtrees                 :1
  Nr. of Sinks                    :23200
  Nr. of Buffer                   :542
  ```

```
Nr. of Level (including gates)  :8
Max trig. edge delay at sink(R) :FF_27/CK 1340.8(ps)
min trig. edge delay at sink(R) :FF_44/CK 1208.7(ps)

                            (Actual)             (Required)
Rise Phase Delay           :1208.7~1340.8(ps)    0~1000(ps)
Fall Phase Delay           :1218.2~1353.6(ps)    0~1000(ps)
Trig. Edge Skew            :132.1(ps)            50(ps)
Rise Skew                  :132.1(ps)
Fall Skew                  :135.4(ps)
Max. Rise Buffer Tran.     :500(ps)              300(ps)
...
Min. Rise Sink Tran.       :143.6(ps)            0(ps)
Min. Fall Sink Tran.       :136.5(ps)            0(ps)
```

reportClockTreeGateRatio

```
reportClockTreeGateRatio
     [-num number]
     [-clk clkName]
     [-file fileName]
     [-detail]
     [-clkRouteOnly | -postRoute]
```

Analyzes the clock tree structure with respect to delay variation, and reports various gate ratio values and violations.

CTS computes the following global gate ratio, as defined:

- gate ratio = *total cell delay* / *path delay from clock root to register*

CTS also computes the following local gate ratios, as defined:

- fmGateRatio = *total cell delay* / *path delay from branching point to source register*
- toGateRatio = *total cell delay* / *path delay from branching point to destination register*
- DeltaGateRatio = fmGateRatio - toGateRatio

By default, CTS reports maximum gate, minimum gate, and maximum delta gate ratio values and violations, if any violations exist.

To use the `reportClockTreeGateRatio` command, you must first specify the following statements in the clock tree specification file:

- `MaxGateRatio` *`ratio`* (default = 1)
- `MinGateRatio` *`ratio`* (default = 0)
- `MaxDeltaGateRatio` *`delta_ratio`* (default = 1)

The timing constraints file must contain the `create_clock` statement in order for CTS to search for register pairs and compute DeltaGateRatio violations.

Parameters

`-clk` *`clkName`*	Specifies the name of the clock to analyze. *Default:* Analyzes all clocks in the design

`-clkRouteOnly`	Analyzes only clock tree wires, even if other wires (such as signal net wires) exist in the design. *Default:* Reports results based on a Steiner estimation of the clock tree.
`-detail`	Reports detailed path delay information for each path. *Default:* Reports only the summary section information
`-file` *`fileName`*	Writes the report to the specified file. *Default:* Writes the report to the log file
`-num` *`number`*	Specifies the maximum number of paths to report. *Default:* `10`
`-postRoute`	Analyzes both clock tree wires and signal wires. *Default:* Reports results based on a Steiner estimation of the clock tree.

Examples

Assume the clock tree spec file contains the following information:

```
AutoCTSRootPin clk
MinDelay 0
MaxDelay 5ns
...
MaxGateRatio 0.9
MinGateRatio 0.2
MaxDeltaGateRatio 0.05
...
```

- The following commands load the clock tree specification file and create clock trees:

  ```
  specifyClockTree -clkfile clk.ctstch
  ckSynthesis
  ```

- The following command generates a summary report of the gate ratio values and violations for the clock `clk`. The report lists the 10 paths with the most violations:

  ```
  reportClockTreeGateRatio -clk clk
  ```

  The software generates the following report:

  ```
  ##########################################################################
  # Clock Tree Gate Ratio Report
  ##########################################################################
  ```

```
================================================================
**** Summary Report ****
================================================================

Clock                  : clk
Mode                   : preRoute
MaxGateRatio           : 0.800000
MinGateRatio           : 0.400000
MaxDeltaGateRatio      : 1.000000
================================================================

Subtrees                          : 3
Sinks                             : 811
Buffers                           : 170
Adjacent Pairs                    : 8259
Max. Gate Ratio Violations        : 811
Min. Gate Ratio Violations        : 0
Max. Delat Gate Ratio Violations : 0
================================================================

**** Max. Gate Ratio Violation ****

No. GateDelay(ps) PathDelay(ps) GateRatio Violation PinName
----------------------------------------------------------------
0   1671          1674          0.998     0.198
atahost1/u1/dma_control/rxbuf/ram_2p_32x32/aa_s_reg_0/CK
1   1663          1667          0.998     0.198
atahost1/u1/dma_control/rxbuf/ram_2p_32x32/bistctrl_0/r_reg_rd_2/CK
2   1663          1667          0.998     0.198
atahost1/u1/dma_control/rxbuf/ram_2p_32x32/bistctrl_0/r_reg_rd_5/CK
...
9   1670          1674          0.998     0.198
atahost1/u1/pio_control/pio_access_control/pio_timing_controller/
t1_cnt/cnt/qi_reg_5/CKCK
================================================================

**** No Min. Gate Ratio Violation ****

================================================================

**** Max. Delta Gate Ratio Violation ****
```

```
No. fmGateRatio toGateRatio Delta fmPinName/toPinName
-------------------------------------------------------------------
0   0.998       0.995       0.002
atahost1/u1/dma_control/rxbuf/rd_ptr_lfsr/iq_reg_3/CK
        atahost1/u1/dma_control/rxbuf/ram_2p_32x32/ram0/CLKB
1   0.995       0.998       0.002
atahost1/u1/dma_control/rxbuf/ram_2p_32x32/ram0/CLKB
        ambawbwrapper1/hrdata_s_reg_reg_22/CK
...
...
9   0.998       0.996       0.002
atahost1/u1/dma_control/rxbuf/wr_ptr_lfsr/iq_reg_3/CK
        atahost1/u1/dma_control/rxbuf/ram_2p_32x32/ram0/CLKA
```

- The following example shows a section of a detailed clock tree gate ratio report for the clock `clk`. It contains detailed path information for each path (by default, 10 paths).

```
===================================================================
**** Detail Report ****
===================================================================
* Clock      : clk
* Path No.   : 5
* fmPoint    : ram0/CLKB(R)
* fmGateRatio: 0.9985
* toPoint    : hrdata_s_reg_reg_2/CK(R)
* toGateRatio: 0.998
* level(fmPOint/toPoint/branchPoint) : 9/10/5

-------------------------------------------------------------------
Object name                      Delta(ps)     Sum(ps)      Attr
-------------------------------------------------------------------
clk (top_ata)                      0             0            f
clk                                0             0            f
clk__L1_IO A->Y (CLKBUFX4)       205           205            f
...
clk_out___L3_N1                    2          1648            r
ram0/CLKB(RAM2P_128x32)            0          1648            r
-------------------------------------------------------------------
Object name                      Delta(ps)     Sum(ps)      Attr
-------------------------------------------------------------------
```

```
clk (top_ata)                        0            0           f
clk                                  0            0           f
clk__L1_IO A->Y (CLKBUFX4)         205          205           f
...
clk_out___L4_N65              )      0         1649           r
hrdata_s_reg_reg_2/CK(SDFFRHQX4)     0         1649           r
=============================================================================
```

reportClockTreeOCV

```
reportClockTreeOCV
     [-num number]
     [-clk clkName]
     [-file fileName]
     [-detail]
     [-preRoute | -clkRouteOnly | -postRoute]
```

Defines OCV report characteristics.

You can use the `reportClockTreeOCV` command after loading timing constraints. If you do not have timing constraints loaded, the `reportClockTreeOCV` command fails because it cannot find adjacent register pairs.

Parameters

`-clk` *`clkName`*	Specifies a clock root name (as defined in the clock tree specification file) on which to report OCV analysis results. *Default*: If you omit this parameter, CTS reports on all the clocks defined in the clock tree specification file.
`-clkRouteOnly`	Creates delay variation and OCV analysis results that are based only on clock tree wires, even if other wires (such as signal net wires) exist in the design. *Default*: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.
`-detail`	Instructs CTS to provide a detailed OCV report. *Default*: If you omit this parameter, CTS creates a summary report.
`-file` *`fileName`*	Specifies the output file for the OCV information. *Default*: If you omit this parameter, CTS outputs the OCV results in the Encounter console and in a log file. **Note:** If you use this parameter, the report information appears *only* in the file you specify—not on your screen.
`-num` *`number`*	Specifies the number of worst-skew-path pairs for which to report OCV analysis results. Specify `1` or a larger number. *Default*: If you do not specify this parameter, only the worst-skew-path pair is reported.

`-postRoute`	Creates delay variation and OCV analysis results that are based on clock tree wires and signal wires. *Default*: If you omit this parameter, CTS behaves as if you had specified the `-preRoute` parameter.
`-preRoute`	Creates delay variation and OCV analysis results that are based on a Steiner estimation of the clock tree. *Default*: This parameter represents the default behavior of CTS if you do not specify `-preRoute`, `-postRoute`, or `-clkRouteOnly`.

Examples

- The following command illustrates how to create a specific, detailed report that should include up to 100 worst-skew-path pairs:

  ```
  reportClockTreeOCV -num 100 -outfile delay_var.rpt -detail
  ```

routeClockNetWithGuide

```
routeClockNetWithGuide
     [-clk clock_root_pin_name]
     [-skipNano]
```

Generates a routing guide file and instructs NanoRoute to route clock nets based on this file.

When you use the `routeClockNetWithGuide` command, CTS

- Generates wire structures based on the CTS routing guide file
- Uses NanoRoute™ to correct DRC violations

You can use the `routeClockNetWithGuide` command before or after using the `specifyClockTree` command, and after using the `ckSynthesis` command.

Parameters

`-clk clock_root_pin_name`

Generates a routing guide file for the specified clock root pin.

Default: Generates a routing guide file for all root pins defined with `AutoCTSRootPin` statements in the clock tree specification file.

`-skipNano`

Instructs CTS to use Trial Route instead of the NanoRoute router to route clock nets.

Note: Use this parameter *only* for debugging purposes.

saveClockBuffers

```
saveClockBuffers
     [-clk clkName]
     [-def]
     -file fileName
```

Creates a file that reports all the clock buffers inserted in the clock tree.

You can use the `saveClockBuffers` command after performing clock tree synthesis on the design.

Parameters

`-clk clkName`	Specifies the clock root name defined in the clock tree specification file. *Default:* If you omit this parameter, CTS reports on all the clocks in the clock tree specification file.
`-def`	Instructs CTS to create buffer names that are DEF-compatible. If you do not use this parameter, CTS creates buffer names that are Verilog-compatible.
`-file fileName`	Specifies the name of the report file.

Examples

- The following command creates a file reporting all the clock buffers after performing CTS:

  ```
  saveClockBuffers -output ClockBuff.rpt
  ```

saveClockNets

```
saveClockNets
     [-clk clkName]
     [-def]
     -file fileName
```

Creates a file that reports all the clock nets in the clock tree. Use this file with the back-end detailed router to help route the clock tree routes.

You can use the `saveClockNets` command after performing clock tree synthesis on the design.

Parameters

`-clk` *`clkName`*	Specifies the clock root name defined in the clock tree specification file. *Default:* If you omit this parameter, CTS reports on all clocks in the clock tree specification file.
`-def`	Instructs CTS to create net names that are DEF-compatible. If you do not use this parameter, CTS creates net names that are Verilog-compatible.
`-file` *`fileName`*	Specifies the name of the report file.

Examples

- The following command creates a file reporting all the clock nets after CTS:

  ```
  saveClockNets -output ClockNets.rpt
  ```

saveClockTreeSpec

```
saveClockTreeSpec
     -file fileName
```

Saves the clock tree specification file from a completed CTS session.

You can use the `saveClockTreeSpec` command after performing clock tree synthesis on the design (including placement and routing).

Parameters

`-file` *`fileName`*	Specifies the name of the end-of-session clock tree specification file.

Examples

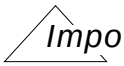
- The following command saves the results from the current CTS session to the file `clk.cts`:

  ```
  saveClockTreeSpec -outfile clk.cts
  ```

setClockNetRouteAttribute

`setClockNetRouteAttribute -file` *`fileName`*

Creates a file—for use with the NanoRoute™ Ultra SoC routing solution—that contains attributes for all clock nets. The attributes are based on information in the clock tree specification file. The attributes are expressed in the output file as `pdi*` commands.



Important

Use this command only if you intend to run NanoRoute Ultra outside the Encounter environment.

You can use the `setClockNetRouteAttribute` command during clock tree synthesis.

Parameters

`-file` *`fileName`*	Specifies the name of the output file that contains the clock net routing attributes.

setCTSMode

```
setCTSMode
     [-help]
     [-reset]
     [-addClockRootProp {true|false}]
     [-fixLeafInst {true|false}]
     [-fixNonLeafInst {true|false}]
     [-honorFence {true|false}]
     [-moveGate {true|false}]
     [-nameAppendParentInClones {true | false}]
     [-nameSingleDelim {true|false}]
     [-opt {true|false}]
     [-optAddBuffer {true|false}]
     [-optLatency {true | false}]
     [-optLatencyMoveGate {true | false}]
     [-powerAware {true|false}]
     [-reportHTML {true|false}]
     [-routeBottomPreferredLayer number]
     [-routeClkNet {true|false}]
     [-routeGuide {true|false}]
     [-routeLeafBottomPreferredLayer number]
     [-routeLeafNonDefaultRule ruleName]
     [-routeLeafPreferredExtraSpace spaceValue]
     [-routeLeafRouteTypeOnGate {true|false}]
     [-routeLeafShielding netName]
     [-routeLeafTopPreferredLayer number]
     [-routeNonDefaultRule ruleName]
     [-routePreferredExtraSpace spaceValue]
     [-routeShielding netName]
     [-routeTopPreferredLayer number]
     [-specMultiMode {true | false}]
     [-traceAsyncSRPinAsLeaf {true | false}]
     [-traceBlackBoxPinAsLeaf {true | false}]
     [-traceDPinAsLeaf {true|false}]
     [-traceHonorConstants {true | false}]
     [-traceIoPinAsLeaf {true|false}]
     [-traceTriStateEnablePinAsLeaf {true | false}]
     [-useLefACLimit {true|false}]
     [-useLibMaxCap {true|false}]
     [-useLibMaxFanout {true|false}]
     [-verbose {true|false}]
```

Sets global parameters for clock tree synthesis. Parameters you specify with the `setCTSMode` command are then used automatically whenever clock tree synthesis is performed. These parameters include default routing attributes that CTS uses if no `RouteType` statement is specified for a clock tree in the clock tree specification file. Settings in the clock tree specification file take priority over settings defined by `setCTSMode`.

Use the `getCTSMode` command to return the current settings for the `setCTSMode` command.

Important

You must set *all* `setCTSMode` parameters before running the `specifyClockTree` command. If you want to change `setCTSMode` parameters after running the `specifyClockTree` command, you must use the `cleanupSpecifyClockTree` command before you specify the `setCTSMode` command again.

Parameters

`-addClockRootProp {true | false}`

Adds the `CLOCKROOT` property to each clock that CTS inserts in a design.

Default: `false`

`-fixLeafInst {true | false}`

Changes the placement status of leaf instances to `FIXED` or `PLACED`.

Default: `true`

`-fixNonLeafInst {true | false}`

Changes the placement status of non-leaf instances to `FIXED` or `PLACED`.

Default: `false`

`-help`

Outputs a brief description that includes type and default information for each `setCTSMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setCTSMode`.

`-honorFence {true | false}`

Controls whether CTS enters more than once into a fence in hierarchical designs.

If you specify `true`, CTS does not create any extra ports for fences.

Default: `false` (This means that CTS might create extra ports.)

`-moveGate {true | false}`

Moves a driving gate during synthesis.

Default: `true`

`-nameAppendParentInClones {true | false}`

Appends the parent instance or net name to clones created with the `ckCloneGate` command.

If you specify `-nameAppendParentInClones false`, clones are named using the following convention:

`FECTS_CLONE_I#` (for cloned instances)

`FECTS_CLONE_N#` (for nets created by `ckCloneGate`)

If you specify `-nameAppendParentInClones true`, clones are named using the following convention:

`original_instance_name``_FECTS_CLONE_I#`

`original_net_name``_FECTS_CLONE_N#`

Default: `false`

`-nameSingleDelim {true | false}`

Uses single name delimiters after the first element in clock root and net names.

CTS normally inserts double (or, in some cases, multiple) name delimiters after the first element of clock root or net names. If you specify `-nameSingleDelim true`, CTS inserts a single underscore (_) in clock root and net names.

Default: `false`

`-opt {true | false}`

Resizes buffers or inverters, refines placement, and corrects routing for signal and clock wires.

If you specify `-opt true`, CTS *attempts* to resize gating components; however, it might not do so. If you specify `-opt false`, CTS does *not* resize gating components.

Default: `true`

`-optAddBuffer {true | false}`

Adds buffers during optimization.

Note: You must specify `-opt true` in order to use this parameter.

Default: `false`

`-optLatency {true | false}`

Reduces the latency during post-optimization stage of CTS by moving, resizing, deleting, and splitting the buffers when set to `true`. It also minimizes the skew during clock tree synthesis.

Default: `false`

`-optLatencyMoveGate {true | false}`

Allows the tool to move the gating cell for latency reduction during clock tree synthesis.

When `optLatency` and `optLatencyMoveGate` are set to `true`, the tool moves the gate in the post-optimization stage.

Note: The `optLatency` parameter must be set to `true` before honoring the `optLatencyMoveGate` parameter.

Default: `true`

`-powerAware {true | false}`

Considers internal power, leakage power, and switching power (based on cv^2f) when building clock trees. Power reporting is enabled with `-powerAware true`.

Note: You must specify the `Period` statement in the clock tree specification file if you use `-powerAware true`.

Default: `false`

`-reportHTML {true | false}`

Generates HTML versions of CTS reports.

Note: Specifying either setting for this parameter will override the normal reporting behavior of the `ckECO`, `ckSynthesis`, and `reportClockTree` commands.

Default: `false`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setCTSMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-routeBottomPreferredLayer` *`number`*

Specifies the bottom preferred metal layer for routing non leaf-level nets.

Default: 3

`-routeClkNet {true | false}`

Routes clock nets.

Default: `false`

`-routeGuide {true | false}`

Determines whether Nanoroute uses the CTS-generated route guide file when routing clock nets. The route guide marks routable gcells for the trunk portion of the net, and provides a list of pins per clock net that should be connected first, to maintain a balanced route topology.

For information on how to use this parameter in combination with the `RouteClkNet` clock tree specification file statement, see “Synthesizing Clock Trees” in the *Encounter Use Guide*.

Default: `true`

`-routeLeafBottomPreferredLayer` *`number`*

Specifies the bottom preferred metal layer for routing leaf-level nets.

Default: 3

`-routeLeafNonDefaultRule` *`ruleName`*

Specifies the LEF `NONDEFAULTRULE` statement that the router should use for routing leaf-level nets.

If you specify this parameter without a rule name, CTS removes any previously specified leaf nondefault rules.

Default: `“ ”` (empty string)

`-routeLeafPreferredExtraSpace` *`spaceValue`*

Specifies the extra space (in tracks) to add around clock wires, when routing leaf-level nets. This parameter matches `setAttribute -preferred_extra_space`, which controls pitch spacing. Both NanoRoute and Trial Route honor this attribute.

Default: `1`

`-routeLeafRouteTypeOnGate {true | false}`

Assigns either `RouteType` or `LeafRouteType` for nets that connect to both leaf pins and gating pins.

Default: `true`

`-routeLeafShielding` *`netName`*

Specifies the ground net name for routing leaf-level nets.

If you specify this parameter without a ground net name, CTS removes any previously specified ground nets.

Default: `" "` (empty string)

`-routeLeafTopPreferredLayer` *`number`*

Specifies the top preferred metal layer for routing leaf-level nets.

Default: `4`

`-routeNonDefaultRule` *`ruleName`*

Specifies the LEF `NONDEFAULTRULE` statement that the router should use for routing non leaf-level nets.

If you specify this parameter without a rule name, CTS removes any previously specified nondefault rules.

Default: `" "` (empty string)

`-routePreferredExtraSpace` *`spaceValue`*

Specifies the extra space (in tracks) to add around clock wires, when routing non leaf-level nets. This parameter matches `setAttribute -preferred_extra_space`, which controls pitch spacing. Both NanoRoute and Trial route honor this attribute.

Default: `1`

`-routeShielding` *`netName`*

Defines the ground net name for routing non leaf-level nets.

If you specify this parameter without a ground net name, CTS removes any previously specified ground nets.

Default: `" "` (empty string)

`-routeTopPreferredLayer` *`number`*

Specifies the top preferred metal layer for routing non leaf-level nets.

Default: 4

```
-specMultiMode {true | false}
```

Enables the new clock tree specification file generator, which generates a specification file that is more timing aware. This parameter can be used only with the `clockDesign` command.

When you specify the following commands when the software is not in multi-mode multi-corner (MMMC) analysis mode:

```
setCTSMode -specMultiMode true
addCTSCellList “BUFX2 BUFX4 BUFX8”
clockDesign -genSpecOnly
```

The auto clock tree spec generator determines whether certain groups of flops that form a clock divider circuit need to be balanced separately. Additionally, it handles case analysis constraints, disabled timing, and loop breaking in the SDC file. It also:

- Generates a spec file that will synthesize separate clock trees for instances where the flops driven by the generated clock do not communicate with the flops driven by the master clock.
- Adds dynamic macromodels when appropriate.
- Groups clocks that communicate with each other in the spec file (`ClkGroup`) when appropriate.

If you specify the same commands when the software is in MMMC analysis mode, the auto clock tree spec generator generates multiple specification files as above, using the naming convention *`fileName.modeName`*. It also performs the following CTS functions on all active analysis views,sequentially:

- Creates and loads a clock tree specification file for each mode.
- Runs clock tree synthesis.
- Determines which instances of the synthesized clock tree to preserve for building the next clock tree.
- Removes the clock tree spec file from memory.

After the software runs clock tree synthesis on all of the views, it runs `timeDesign`.

Default: `false`

`-synthLatencyEffort {high | low}`

Puts more effort during initial clock tree construction in order to achieve a clock tree that has less clock insertion delay. When set to `high`, the maximum fanout constraint is ignored.

Default: `low`

`-traceAsyncSRPinAsLeaf {true | false}`

Treats asynchronous set and reset pins as leaf pins.

Default: `false`

`-traceBlackBoxPinAsLeafPin {true | false}`

Treats input pins of black boxes (macro blocks that do not have timing libraries, or timing arcs from the pins), as leaf pins.

Default: `false`

`-traceDPinAsLeaf {true | false}`

Treats the Data pins of flip-flops as synchronous pins, instead of as excluded pins.

Data pins include:

- Data pins
- Enable pins
- Scan-in pins
- Scan-enable pins[1]
- Synchronous set and reset pins

This parameter does not control tristate control pins. By default, the software treats them as synchronous pins.

Default: `false`

`-traceHonorConstants {true | false}`

Honors the following SDC constraints captured in the memory after reading the timing constraints file, and propagates the constants when tracing the clock tree.

- `set_case_analysis`
- `set_disable_timing`
- `set_logic_one`
- `set_logic_zero`
- 1’b0 and 1’b1 in the netlist

When set to `false`, the software ignores the constraints when tracing the clock tree. For example, it traces through an AND gate, where the non-clock input pin has a `set_case_analysis 0`.

When set to `true`, the software stops propagating constants at the AND gate, and marks the clock input pin as an excluded pin.

Default: `false`

`-traceIoPinAsLeaf {true | false}`

Treats I/O pins as synchronous pins, instead of as excluded pins.

This parameter does not control tristate control pins. By default, the software treats them as synchronous pins.

Default: `false`

`-traceTriStateEnablePinAsLeaf {true | false}`

Treats enable pins of tristate instances as leaf pins.

Default: `false`

`-useLefACLimit {true | false}`

Considers electromigration (EM) during clock tree synthesis.

If you specify `-useLefACLimit true`, CTS tries to minimize EM violations during synthesis by resizing or inserting buffers to enhance the electrical current flow through the wires. If any wire exceeds the library value, CTS reports the violation in the standard CTS report.

In order to perform EM analysis and optimization, your LEF file must contain an AC Limit table. You must also specify the `Period` statement in the clock tree specification file. You can use `clockDesign -genSpecOnly` to convert the period value from the SDC constraints.

Default: `false`

`-useLibMaxCap {true | false}`

Uses the maximum capacitance values specified in the timing library.

If you specify `-useLibMaxCap true` and there are no `PinMaxCap`, `MaxCap`, or `DefaultMaxCap` statements specified in the clock tree specification file, the software attempts to correct maximum capacitance violations based on the maximum capacitance values specified in the timing library for buffers, inverters, gating cells, and pins.

If you specify `-useLibMaxCap true` and there is a `PinMaxCap`, `MaxCap`, or `DefaultMaxCap` statement specified in the clock tree specification file, the maximum capacitance values specified in the specification file (in the statement order listed) override the values specified in the timing library.

Default: `false`

`-useLibMaxFanout {true | false}`

Uses the maximum fanout constraints in the timing library.

If you specify `-useLibMaxFanout true` and there is no `MaxFanout` statement specified in the clock tree specification file, the software limits the number of clock buffer fanouts to the value specified in the timing library.

If you specify `-useLibMaxFanout true` and there is a `MaxFanout` statement specified in the clock tree specification file, CTS uses the worst case constraint specified.

CTS does not support a `fanout_load` that is not equal to 1.

Note: CTS considers `-useLibMaxFanout` to be a soft constraint. CTS will try to meet it, but might need to make adjustments in order to meet physical constraints.

Default: `false`

`-verbose {true | false}`

Controls the amount of information displayed in the Encounter console and written to the log files during CTS.

Default: `false`

1.

Examples

- The following command instructs CTS add the `CLOCKROOT` property for different clocks in a design during clock tree synthesis:

  ```
  setCTSMode -addClockRootProp true
  ```

- The following command instructs CTS to route clock nets furing clock tree synthesis:

  ```
  setCTSMode -routeClkNet true
  ```

- The following command resets the `-routeClkNet` and `-powerAware` parameters to their default values:

  ```
  setCTSMode -reset -routeClkNet -powerAware
  ```

- The following command resets all `setCTSMode` parameters to their default settings:

  ```
  setCTSMode -reset
  ```

Related Topics

- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*

specifyClockTree

```
specifyClockTree
     {-file fileName | -template}
     [-dont_use]
```

Loads the clock tree specification file. You can also use this command to create a clock tree specification template file. Your design does not need to be placed for you to use the `specifyClockTree` command.

When CTS is in multi-corner mode, the `specifyClockTree` command reports the active analysis views on which it will operate. It also lists the RC information for each active view, including the estimated capacitance, resistance, and via resistance for each view.

Multi-corner CTS does not support the timing constraints file in the clock tree specification file.

Parameters

`-dont_use`	Excludes buffers from the clock tree specification file that are defined as `dont_use : true` in the `.lib` file. *Default*: `specifyClockTree` uses all buffers defined in the clock tree specification file.
`-file` *`fileName`*	Specifies the name of the clock tree specification file.
`-template`	Creates a sample clock tree specification template file in the directory in which you started the Encounter software. The filename is `template.ctstch`.

Loading Multi-Corner Macro Model Files

When CTS is in multi-corner mode, the `specifyClockTree` command can load a multi-corner macro model file. CTS interprets multi-corner macro model information in the following way:

- Any macro model statement in the file that does not have a view name applies to all unspecified views.

For example:

Assume there are the following three active views in the design: `view1`, `view2`, `view3`. In multi-corner mode, if the macro model file contains the following macro model statements:

```
MacroModel port block/clk 100ps 100ps 100ps 100ps 10ff
```

CTS interprets the information as:

```
MacroModel port block/clk 100ps 100ps 100ps 100ps 10ff view1
MacroModel port block/clk 100ps 100ps 100ps 100ps 10ff view2
MacroModel port block/clk 100ps 100ps 100ps 100ps 10ff view3
```

If the macro model file contains the following macro model statements:

```
MacroModel port block/clk 100ps 100ps 100ps 100ps 10ff view1
MacroModel port block/clk 200ps 200ps 200ps 200ps 20ff view2
```

CTS interprets the information as:

```
MacroModel port block/clk 100ps 100ps 100ps 100ps 10ff view1
MacroModel port block/clk 200ps 200ps 200ps 200ps 20ff view2
MacroModel port block/clk 200ps 200ps 200ps 200ps 20ff view3
```

Including Buffers In A Clock Tree Specification File

Sometimes a designer includes buffers in a clock tree specification file that are defined as `dont_use : true` in the `.lib` file.

Clock tree specification file syntax

```
AutoCTSRootPin      CK
MaxDelay            1600ps
MinDelay            1400ps
SinkMaxTran         200ps
BufMaxTran          300ps
MaxSkew             200ps
NoGating            NO
Buffer              BUF1 BUF2 BUF3
```

Library file syntax

```
cell (BUF1) {
          area : 200.000000 ;
          dont_touch : true ;
          dont_use : true ;

cell (BUF2) {
          area : 100.000000 ;
          dont_touch : true ;
```

```
          dont_use : true ;

cell (BUF3) {
          area : 50.000000 ;
          dont_touch : true ;
          dont_use : true ;
```

Examples

- The following command loads the `design_32.cts` clock tree specification file:

  ```
  specifyClockTree -clkfile design_32.cts
  ```

- The following example shows the active analysis views and RC information sections reported in a clock tree specification file that was loaded when CTS was in multi-corner mode:

  ```
  Active Views for CTS:
  #1 view1
  #2 view2
  #3 view3
  ...
  RouteType             : FE_CTS_DEFAULT
  PreferredExtraSpace   : 1
  Shield                : NONE
  PreferLayer           : M3 M4
  ...
  RC Information for View view1:
  Est. Cap              : 0.1377(V=0.1377 H=0.1377)(ff/um)[0.0001377]
  Est. Res.             : 1.43466(V=1.43466 H=1.43466)(ohm/um)[0.000143466]
  Est. Via Res          : 6.71366(ohm)[13.7071]
  M1(H)w=0.14(um)s=0.12(um)p=0.28(um)es=0.42(um)cap=0.15(ff/um)res=1.11(ohm/
  um)via=0(ohm)
  M2(V)w=0.14(um)s=0.14(um)p=0.28(um)es=0.42(um)cap=0.138(ff/um)res=1.43(ohm/
  um)via=6.99339(ohm)
  M3(H)w=0.14(um)s=0.14(um)p=0.28(um)es=0.42(um)cap=0.138(ff/um)res=1.43(ohm/
  um)via=6.71366(ohm)
  ...
  RC Information for View view2:
  Est. Cap              : 0.1177(V=0.1177 H=0.1177)(ff/um)[0.0001177]
  Est. Res.             : 1.33466(V=1.33466 H=1.33466)(ohm/um)[0.000133466]
  Est. Via Res          : 2.1(ohm)[4.2875]
  M1(H)w=0.14(um)s=0.12(um)p=0.28(um)es=0.42(um)cap=0.13(ff/um)res=1.10(ohm/
  um)via=0(ohm)
  ```

```
M2(V)w=0.14(um)s=0.14(um)p=0.28(um)es=0.42(um)cap=0.128(ff/um)res=1.33(ohm/
um)via=2.1875(ohm)
M3(H)w=0.14(um)s=0.14(um)p=0.28(um)es=0.42(um)cap=0.128(ff/um)res=1.33(ohm/
um)via=2.1(ohm)
...
```

Related Topics

- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

4

Power Calculation Commands

dump_unannotated_nets

```
dump_unannotated_nets
     [-file filename]
     -type annotationType
```

Shows the percentage of nets annotated by a specific type and also prints the names of all nets that are not annotated by that type.

Annotation information is stored on a per-file basis for activity, VCD, TCF, and user commands to provide detailed annotation reports when requested.

`dump_unannotated_nets` can be invoked multiple times with different report types (`-type`).

Parameters

`-file` *`filename`*

Specifies the text file name that includes the annotation report information. If no file is specified, the report will be written to the default output file `annotation_`*`type`*`.log`, where *`type`* is taken from the `-type` argument specified.

`-type` *`annotationType`*

The type can be any of the following:

- `all`

  Shows annotation reports for all types.

- `activity`

  Shows annotation data for all types of activity specifications including TCF, VCD, SDC, TWF, and user commands that set activity.

- `vcd`

  Shows annotation from VCD files.

- `tcf`

  Shows annotations from TCF files.

get_power_analysis_mode

```
get_power_analysis_mode
     [-help]
     [-quiet]
```

Returns the following information about a set_power_analysis_mode parameter in the log file and in the software console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software returns information for all of the set_power_analysis_mode parameters.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `get_power_analysis_mode` parameter. For a detailed description of the command and all of its parameters, use the man command `man get_power_analysis_mode`.
`-quiet`	Returns the current settings of all set_power_analysis_mode parameters in Tcl list format.

Example

- The following command returns the current settings for all set_power_analysis_mode parameters:

```
get_power_analysis_mode \
-honor_negative_energy true        # bool, default=true \
-ignore_control_signals true       # bool, default=true \
-leakage_scale_factor_for_temp 1   # float, default=1 \
-off_pg_nets {}                    # string, default="" \
-x_transition_factor 0.5           # float, default=0.5 \
-z_transition_factor 0.25          # float, default=0.25
```

- The following command returns the current settings for all set_power_analysis_mode parameters in Tcl list format only:

  ```
  get_power_analysis_mode -quiet
  ```

  The software returns the following information:

  ```
  {honor_negative_energy true} {ignore_control_signals true}
  {leakage_scale_factor_for_temp 1} {off_pg_nets {}} {x_transition_factor 0.5}
  {z_transition_factor 0.25}
  ```

map_activity_file

```
map_activity_file
     [-help]
     -golden [rtl | gate]
     -rtl2gate mapping_file
```

Specifies instance name mapping between RTL netlist and GATE level netlist. Power analysis uses this instance name mapping to annotate the output nets of this instance from the pin based RTL VCD or TCF.

Parameters

`-golden {RTL | GATE}`

Specifies whether RTL or Gate is Golden. The one not specified as golden will be considered as Revised. Currently the Conformal MatchPoint file does not specify this.

Default: `RTL`

`-help`

Outputs a brief description that includes the type and default information for each `map_activity_file` parameter.

For a detailed description of the command and all of its parameters, use the man command `man map_activity_file`.

`-rtl2gate` *`mapping_file`*

Specifies instance name mapping between RTL netlist and GATE level netlist. RTL level VCD or TCF are obtained from simulation on RTL Verilog netlist. Power analysis uses this instance name mapping to annotate the output nets of this instance from the pin based RTL VCD or TCF.

Example

- The following command specifies a mapping file `rtl_map`:

  ```
  map_activity_file -rtl2gate rtl_map
  ```

- The following command performs instance name mapping between RTL netlist and GATE level netlist, wherein the GATE level netlist is Golden and the RTL netlist is Revised; it specifies the mapping file `MapFile.alt`

  ```
  map_activity_file -rtl2gate MapFile.alt -golden gate
  ```

An example of the mapping file output from Conformal is described as follows:

```
Mapped points: SYSTEM class
1-th mapped points:
  (G) + 1      PI   /x_ick
  (R) + 595    PI   /x_ick
2-th mapped points:
  (G) + 2      PI   /x_jreset_cp_p
  (R) + 594    PI   /x_jreset_cp_p
3-th mapped points:
  (G) + 3      PI   /x_mreset_cp_p
  (R) + 593    PI   /x_mreset_cp_p
........
60-th mapped points:
  (G) + 4582   DFF  /cpexec0/cpddecls0/gi_opls/q_reg_reg[35]
  (R) + 3856   DFF  /cpexec0/cpddecls0/gi_opls/q_reg_reg_35_
61-th mapped points:
  (G) + 4583   DFF  /cpexec0/cpddecls0/gi_opls/q_reg_reg[34]
  (R) + 3855   DFF  /cpexec0/cpddecls0/gi_opls/q_reg_reg_34_
62-th mapped points:
  (G) + 4633   DFF  /cpexec0/cpddecex0/gi_opex/q_reg_reg[21]
  (R) + 3805   DFF  /cpexec0/cpddecex0/gi_opex/q_reg_reg_21_
63-th mapped points:
  (G) + 4634   DFF  /cpexec0/cpddecex0/gi_opex/q_reg_reg[20]
  (R) + 3804   DFF  /cpexec0/cpddecex0/gi_opex/q_reg_reg_20_
64-th mapped points:
  (G) + 4635   DFF  /cpexec0/cpddecex0/gi_opex/q_reg_reg[19]
  (R) + 3803   DFF  /cpexec0/cpddecex0/gi_opex/q_reg_reg_19_
65-th mapped points:
  (G) + 4636   DFF  /cpexec0/cpddecex0/gi_opex/q_reg_reg[18]
  (R) + 3802   DFF  /cpexec0/cpddecex0/gi_opex/q_reg_reg_18_
```

where, G refers to the Golden netlist name and R refers to the Revised netlist name.

By default, the tool assumes that the Golden netlist name refers to the RTL net name and Revised netlist refers to the GATE level net name.

propagate_activity

```
propagate_activity
```

Propagates the activity file in the database after it is read in using the read_activity_file command. This command propagates the activity for all primary inputs, nets, and other devices in the design whose activity has not been previously defined through user attributes.

Parameters

None

Example

- The following commands read and propagate the activity file in the database:

```
read_activity_file
propagate_activity
```

read_activity_file

```
read_activity_file file
     [-end time]
     [-format { VCD | TCF | SAF }]
     [-hier_separator separator]
     [-report_missing_nets { true|false }]
     [-reset]
     [-start time]
     [-tcf_block tcf_block_name]
     [-tcf_scope tcf_scope_name]
     [-vcd_block vcd_block_name]
     [-vcd_scope vcd_scope_name]
```

Specfies the name and type of activity file to be used as input.

Parameters

`-end {time1 time2 ...timen}`

When you use the vector-driven method of performing dynamic instance-based power-consumption calculation, you can specify the part of the activity file from which you want power analysis to use for the power calculation. This option specifies the end time of this range and is used in conjunction with the `-start` option.

Specifies the end time as time-value pairs to report the average power across non-overlapping multiple windows specified in the activity file for a block, level of hierarchy, or other part of the design that you want to analyze. Units are in seconds (`s`), milliseconds (`ms`), microseconds (`us`), nanoseconds (`ns`), or picoseconds (`ps`).

Default: The time unit is defined in the first `.lib` file read during design import.

Not required for TCF and SAF.

`-format {VCD|TCF|SAF}`

Specifies the net activity of the design using an activity file in VCD, TCF, or SAF format.

- The VCD file format uses the toggle count information as the basis for the power consumption calculation. These files can be in compressed (zipped) format using the `.gz` extension.
- The TCF file format contains data on the switching activity of the nets in the design. The switching activity includes the toggle count information and the probability of the net or pin being in the logic 1 state.

  These files can be in compressed (zipped) format using the `.gz` extension.

  Supports extended TCF which contains transition density for X and Z as shown in the following extended TCF example:

  ```
  "out" :  "0.0  0"  "0.005000  2"  "0.0  0"
  ```

  The fields are defined as follows:

  Field 1: net or pin name
  Field 2: transition density and duty for transition of 0-1
  Field 3:transition density and duty for transition of X(0-X-1)
  Field 4: transition density and duty for transition of Z(0-Z-1)
- The SAF file format contains switching activity information and one default clock frequency. The switching activity is in terms of a percentage that is relative to the default clock frequency.

`-hier_separator` *`separator`*

Specifies the separator character in the hierarchical net names, bus names, and pin names for the block or other part of the design that you want to analyze. It must be the same as the separator character in the design netlist.

Default: slash (`/`)

`-report_missing_nets {true | false}`

If set to `false`, the missing nets will not be reported in the log file.

Default: `false`

`-reset`

If `-reset` is specified, all previous activity files specified using the `read_activity_file` command will be reset and ignored.

Default: All the previous activity files specified using the `read_activity_file` command are loaded and used (They are additive).

`-start {`*`time1 time2 ... timen`*`}`

When you use the vector-driven method of performing dynamic instance-based power-consumption calculation, you can specify the part of the activity file from which you want power analysis to use for the power calculation. This option specifies the start time of this range and is used in conjunction with the `-end` option.

Specifies the start time as time-value pairs to report the average power across non-overlapping multiple windows specified in the VCD file for a block, level of hierarchy, or other part of the design that you want to analyze. Units are in seconds (`s`), milliseconds (`ms`), microseconds (`us`), nanoseconds (`ns`), or picoseconds (`ps`).

Default: The time unit is defined in the first `.lib` file read during design import.

Not required for TCF and SAF.

`-tcf_block` *`tcf_block_name`*

Specifies the name of the tcf block to map a sub-block activity file with a top level Verilog file. Provides the ability to support multiple activity files based upon block instance names.

The block argument is added to the TCF name before comparing it to the Verilog file.

`-tcf_scope` *`tcf_scope_name`*

Specifies the tcf scope, that is, the name of the module within the activity file associated with the block or other part of the design that you want to analyze.

The scope argument is removed from the TCF name before comparing it to the Verilog file.

`-vcd_block` *`vcd_block_name`*

Specifies the name of the vcd block to map a sub-block activity file with a top level Verilog file. Provides the ability to support multiple activity files based upon block instance names.

The block argument is added to the VCD name before comparing it to the Verilog file.

`-vcd_scope` *`vcd_scope_name`*

Specifies the vcd scope, that is, the name of the module within the activity file associated with the block or other part of the design that you want to analyze.

The scope argument is removed from the VCD name before comparing it to the Verilog file.

`file` The name of the activity file.

Activity Precedence

The following commands can define activity:

```
read_activity_file file
    [-format { VCD | TCF | SAF ]}
    [-hier_separator separator]
    [-start time]
    [-end time]
    [-reset]
    [-vcd_scope vcd_scope_name]
    [-vcd_block vcd_block_name]
    [-tcf_scope tcf_scope_name]
    [-tcf_block tcf_block_name]
    [-report_missing_nets { true|false }
set_default_switching_activity
    [-input_activity factor]
    [-seq_activity factor]
    [-period value]
    [-duty value]
    [-global_activity factor]
```

```
    [-hier hierarchy_name]
    [-clock_gates factor]
set_switching_activity
    [-reset]
    [-clock clock_name]
    [-period value]
    [-unclocked]
    [-activity factor | -density transition_density ]
    [-net net_name | -port port_name | -pin pin_name ]
    [-duty value]
    [-inst]
```

The precedence of activity is as follows:

1. User-Defined: [*net* | *pin* | *instance*] [*activity* | *td*]
2. Activity File: *file scope start stop*
3. Hierarchical Global Activity: *hier_name activity*
4. Global Activity: *activity*
5. Sequential Element Activity: *activity*
6. Clock Gates Enable Activity: *activity*
7. Primary Input Activity: *activity*

If you specify multiple activity data for the same items, the power engine will use the one with the higher precedence.

Examples

- The following command reads the compressed `ap_wait_test_pll1_mod.vcd.gz` VCD file that contains the toggle count information for the power consumption calculations, specifies the `crm_ap` module within the VCD file as the module associated with the part of the design being analyzed, specifies the slash character (`/`) as the hierarchical separator for hierarchical net names, bus names, and pin names for the part of the design being analyzed, and specifies the start (`180002ns`) and end (`189802ns`) times from the VCD file for the part of the design being analyzed:

  ```
  read_activity_file -format VCD \
      -vcd_scope testbench/top/ap/mcu_platform/crm_ap \
      -hier_separator \
      -start 180002ns \
      -end 189802ns \
      ap_wait_test_pll1_mod.vcd.gz
  ```

- The following command reads the compressed `A.vcd.gz` VCD file that contains the toggle count information for the power consumption calculations, specifies `A1` as the block to map a sub-block VCD file with a block instance at the top-level, and specifies the start (`180002ns`) and end (`189802ns`) times from the VCD file for the part of the design being analyzed:

```
read_activity_file -format VCD \
    -vcd_block ap/A1 \
    -start 180002ns \
    -end 189802ns \
    A.vcd.gz
```

- The next example specifies a vcd activity file called `dmac_mac.vcd` for the scope `top/dma_dut`.

```
read_activity_file format vcd start 10ns stop 20ns \
    -vcd_scope top/dma_dut dmac_mac.vcd
```

report_power

```
report_power
     [-cell {cell_list}]
     [-cell_type {all | {macro io combinational sequential clock}}]
     [-clock_network {all | {clock_list}}]
     [-count_seq_elements_in_clock_network]
     [-instances {instance_list}]
     [-hierarchy {all | hierarchy_level}]
     [-leakage]
     [-net [-nworst]]
     [-outfile filename]
     [-pg_net {all | pg_net_name_list}]
     [-power_domain {all | {power_domain_list}}]
     [-rail_analysis_format {FE | VS}]
     [-sort {internal | switching | leakage | total}]
     [-threshold value]
     [-view view_name]
```

Reports global as well as specific instance power, clock network power, clock domain power, and net switching power. It also reports the power for power domains and specific power nets. Units are reported in milliwatts.

Parameters

`-cell {cell_list}`
: Specifies the cells to include in the power report. Accepts wildcards (*)

`-cell_type {all | {macro io combinational sequential clock}}`
: Specifies the cell type to include in the power report.

  Default: `all`

`-clock_network {all | {clock_list}`
: Reports the power consumption of the clock network, including the power for generated clocks.

  Use `clock_list` to report the power consumed by specific clocks.

  Default: `all`

`-count_seq_elements_in_clock_network`

Includes the leaf flip-flop power in the clock network power report.

Default: The leaf flip-flop power will not be included in the clock network power report.

`-hierarchy {all | hierarchy_level}`

Specifies the hierarchy level that is included in the power report. You can specify the *`hierarchy_level`* as a number that corresponds to a specific hierarchy level in the design. A *`hierarchy_level`* of 0 specifies the top level.

Default: `all`

`-instances {instance_list}`

Specifies the instances to include in the power report. The power report provides the amount of power consumed by a specified list of hierarchical or leaf-level instances. *`instance_list`* accepts wildcards (*) for leaf-level instances only.

`-leakage` Reports only leakage power of the design.

`-net` Reports the net switching power, as well as the load capacitance, toggle rate, and switching values for each net in the design.

`-nworst` Reports the number of nets with the highest net switching power.

Note: You must use the `-nworst` parameter in conjunction with the `-net` parameter.

`-outfile filename` Specifies the name of the report file in which the top-level summary of the power consumption information is written. You can include the directory name.

Note: If you do not specify this parameter, the software directs the power report to the command line, as well as the log file, and no power report is written to a separate file.

`-pg_net {all | pg_net_name_list}`

When set to `all`, reports the amount of power consumed by all power nets.

If you specify a list, reports the power consumed by each instance that is connected to the specified list of power nets.

Note: When using this parameter, the static power calculation engine always performs propagation and power calculation.

Default: `all`

`-power_domain {all | {power_domain_list}}`

Specifies all power domains or lists power domains to be included in report.

`-rail_analysis_format {FE | VS}`

Specifies the power file format that is read by the power-grid analysis tool.

FE: Specifies the power file that Encounter TS uses to perform power-rail analysis.

VS: Specifies the power file that VoltageStorm uses to perform power-rail analysis.

`-sort {internal | switching | leakage | total}`

Sorts the instance-based power consumption data by `internal`, `switching`, `leakage`, or `total` power.

Default: `total`

`-threshold value` Filters out the instances whose total power dissipation is less than the specified value. Units are in milliwatts.

Default: `0`

`-view` *`view_name`*	Specifies the view that was created using the multi-mode multi-corner (MMMC) `set_analysis_view` command. Power analysis is performed for the specified view. If `report_power -view` *`view_name`* is set, it will override the view set in either `set_power_analysis_mode -view` or `set_analysis_view` command. *Default*: Uses current view in Encounter.

Examples

- The following command reports the amount of power consumed by the clock nets `clk` and `test_clk` (including leaf flip-flop power) and writes the output information to a power report file named `clk_pwr.rep`:

```
report_power -clock_network{clk test_clk} \
    -count_seq_elements_in_clock_network -outfile clk_pwr.rep
```

- The following command reports the amount of power consumed down to hierarchy level 3 from the top and writes the output information to a power report file named `hier.rep`:

```
report_power -hierarchy 3 -outfile hier.rep
```

- The following command reports the amount of power consumed by leaf-level instances that match `ECO*` and writes the output information to a power report file named `inst.rep`:

```
report_power -instances ECO* -outfile inst.rep
```

- The following command reports the amount of power consumed by all cells that match `buf*6` and writes the output information to a power report file named `cell.rep`:

```
report_power -cell buf*6 -outfile cell.rep
```

- The following command reports the amount of power consumed by all I/O and sequential cells and writes the output information to a power report file named `cell_type.rep`:

```
report_power -cell_type {io sequential} -outfile cell_type.rep
```

- The following command sorts the instance-based power consumption data by `leakage` power and writes the output information to a power report file named `sort.rep`:

```
report_power -sort leakage -outfile sort.rep
```

- The following command reports the total power of all instances whose total power dissipation is less than `1` milliwatt and writes the output information to a power report file named `threshold.rep`:

```
report_power -threshold 1 -outfile threshold.rep
```

- The following command reports the net switching power, load capacitance, toggle rate, and switching values for each net in the design, reports the top `100` nets with the highest net switching power, and writes the output information to a power report file named `net.rep`:

  ```
  report_power -net -nworst 100 -outfile net.rep
  ```

- The following command generates an instance power file named `rail_analysis.rep` that the software can use to perform power-rail analysis using the `updatePower` command:

  ```
  report_power -rail_analysis_format FE -outfile rail_analysis.rep
  ```

- The following command reports the amount of power consumed by each power domain and writes the output information to a power report file named `pd_all.rep`:

  ```
  report_power -power_domain all -outfile pd_all.rep
  ```

- The following command reports the amount of power consumed by each power domain PD2 and writes the output information to a power report file named `pd.rep`:

  ```
  report_power -power_domain PD2 -outfile pd.rep
  ```

- The following command reports the amount of power consumed by view `view_max-hp-max-lp1` and writes the output information to a power report file named `view.rep`:

  ```
  report_power -view view_max-hp-max-lp1 -outfile view.rep
  ```

- The following command reports the amount of power consumed by each instance connected to power nets `vdd` and `vdd1` and writes the output information to a power report file named `pg_net.rep`:

  ```
  report_power -pg_net {vdd vddl} -outfile pg_net.rep
  ```

- Example of the output of the power report.

  ```
  *-----------------------------------------------------------------------
  *        -  - Version  64-bit
  *       Date & Time:    2008-Jan-27 15:49:51 (2008-Jan-27 23:49:51 GMT)
  *-----------------------------------------------------------------------
  *       Design: dtmf_chip
  *
  *       Liberty Libraries used:
  *
  /sev//DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_rvt_ss_1p08v_125c.lib
  *
  /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_rvt_ss_0p8v_1p08v_125c.lib
  *
  /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_rvt_ss_0p8v_125c.lib
  *
  /sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_lvt_ss_1p08v_125c.lib
  *
  /sevDTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_lvt_ss_0p8v_1p08v_125c.lib
  ```

```
*
/sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_lvt_ss_0p8v_125c.lib
*
/sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_hvt_ss_1p08v_125c.lib
*
/sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_hvt_ss_0p8v_1p08v_125c.lib
*
/sev/DTMF_CHIP/TIMING/max/scmetropmk_cmos10lp_hvt_ss_0p8v_125c.lib
*   /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_rvt_ss_1p08v_125c.lib
*   /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_rvt_ss_0p8v_125c.lib
*   /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_lvt_ss_1p08v_125c.lib
*   /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_lvt_ss_0p8v_125c.lib
*   /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_hvt_ss_1p08v_125c.lib
*   /sev/DTMF_CHIP/TIMING/max/scmetro_cmos10lp_hvt_ss_0p8v_125c.lib
*   /sev/TIMING/max/rom_512x16A_ss_1v08_125c_syn.lib
*   /sev/DTMF_CHIP/TIMING/max/pllclk_slow.lib
*
/sev/DTMF_CHIP/TIMING/max/iometroil_cmos10lp_hvt_ss_1p08v_2p3v_2p5v_125c.lib
*
/sev/DTMF_CHIP/TIMING/max/iometroil_cmos10lp_hvt_ss_0p8v_2p3v_2p5v_125c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_rvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_rvt_ff_1p1v_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_lvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_lvt_ff_1p1v_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_hvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetropmk_cmos10lp_hvt_ff_1p1v_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetro_cmos10lp_rvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetro_cmos10lp_lvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/scmetro_cmos10lp_hvt_ff_1p32v_m40c.lib
*
/sev/DTMF_CHIP/TIMING/min/rom_512x16A_ff_1v32_m40c_syn.lib
*
/sev/DTMF_CHIP/TIMING/min/pllclk_fast.lib
*
/sev/DTMF_CHIP/TIMING/min/iometroil_cmos10lp_hvt_ff_1p32v_2p7v_2p5v_m40c.lib
```

```
*
/sev/DTMF_CHIP/TIMING/min/iometroil_cmos10lp_hvt_ff_1p1v_2p7v_2p5v_m40c.lib
*
*       Power Domain used:
*                Rail:     VDD_AO        Voltage:       1.32
*                Rail: VDD_TDSPCore      Voltage:       1.32
*                Rail: VDD_TDSPCore_R    Voltage:       1.32
*
*       Parasitic Files used:
* /sev/DTMF_CHIP/SPEF/dtmf_chip.decoup.spef
*
*       DEF Files used:
* /sev/DTMF_CHIP/DEF/dtmf_chip.def
*
*       Power Units = 1mW
*       Time Units = 1e-09 secs
*       report_power
-------------------------------------------------------------------------------

Total Power
-------------------------------------------------------------------------------
Total Internal Power:      0.01943(62.6774%)
Total Switching Power:     0.01057(34.0968%)
Total Leakage Power:       0.001(3.2258%)
Total Power:               0.031
-------------------------------------------------------------------------------

Group      Internal      Switching     Leakage       Total  Percentage
Power      Power         Power         Power  (%)
-------------------------------------------------------------------------------
Sequential    0.0006738   0.0002215   4.045e-05   0.0009358     3.019
Macro         0.0004916   1.515e-05   0.0002609   0.0007677     2.477
IO            0.01743     0.009085    0.0003172   0.02683      86.56
Combinational 0.0008324   0.001247    0.0003815   0.002461      7.941
-------------------------------------------------------------------------------
Total 0.01943     0.01057       0.001       0.031         100
-------------------------------------------------------------------------------
Rail                   Voltage   Internal   Switching   Leakage
Total  Percentage     Power     Power      Power        Power  (%)
-------------------------------------------------------------------------------
VDD_AO                   1.32  0.0008238   0.0003404
0.0002694    0.001434       4.625
```

```
VDD_TDSPCore
         1.32   0.001126    0.001022   0.0003931    0.002541       8.197
VDD_TDSPCore_R
         1.32  5.833e-06   3.253e-07   8.371e-07   6.996e-06     0.02257

Clock                    Internal   Switching     Leakage
Total  Percentage
                         Power      Power         Power
Power  (%)
------------------------------------------------------------------------
m_digit_clk               0           0           0           0           0
m_ram_clk                 0           0           0           0           0
m_dsram_clk               0           0           0           0           0
m_spi_clk           1.018e-06    1.09e-06   2.572e-08   2.134e-06    0.006884
m_rcc_clk           1.282e-05   1.513e-05   2.672e-07   2.821e-05     0.09103
m_clk               5.039e-05   7.278e-05   7.658e-07   0.0001239      0.3998
refclk              1.11e-05   3.017e-05   2.488e-07   4.152e-05       0.134
------------------------------------------------------------------------
Total               7.533e-05   0.0001192   1.307e-06   0.0001958      0.6317
------------------------------------------------------------------------
*       Power Distribution Summary:
*  Highest Average Power:     IOPADS_INST/Prefclkip (PBCSCT2B):     0.003727
*  Highest Leakage Power: DTMF_INST/ROM_512x16_0_INST (rom_512x16A): 8.899e-05
*               Total Cap:      2.46221e-10 F
*               Total instances in design:  8525
*               Total instances in design with no power:    24
*               Total instances in design with no activty:    24
*               Total Fillers and Decap:     8
------------------------------------------------------------------------
```

- Next command creates a leakage report (see example report below):

```
report_power -leakage

*------------------------------------------------------------------------
*    Encounter Timing System 08.10-b050_1 (64bit) 08/21/2008 19:23 (Linux 2.6)
*
*
*       Date & Time:    2008-Aug-24 21:34:17 (2008-Aug-25 04:34:17 GMT)
*
*------------------------------------------------------------------------
*
*       Design: dma_mac
```

```
*
*       Liberty Libraries used:
*               DATA/timing_libs//tcbn65lp_LVL_c060217wc0d90d9.lib
*               DATA/timing_libs//tcbn65lpcgwc.lib
*               DATA/timing_libs//tcbn65lplvtwc.lib
*               DATA/timing_libs//tcbn65lpwc.lib
*               DATA/timing_libs//tsdn65lpa1024x32m8f_100a_wc.lib
*               DATA/timing_libs//tsdn65lpa128x16m8f_100a_wc.lib
*               DATA/timing_libs//tcbn65lpcgwc0d72.lib
*               DATA/timing_libs//tcbn65lplvtwc0d72.lib
*               DATA/timing_libs//tcbn65lpwc0d72.lib
*               DATA/timing_libs//tsdn65lpa1024x32m8f_0p84v_wc.lib
*               DATA/timing_libs//tsdn65lpa128x16m8f_0p84v_wc.lib
*
*       Power Domain used:
*               Rail:      VDDlu        Voltage:       0.84
*               Rail:       VDDm        Voltage:       1.08
*               Rail:        VDD        Voltage:       1.08
*
*       Power Units = 1mW
*
*       Time Units = 1e-09 secs
*
*       report_power -outfile Leakage.rep -leakage
*
------------------------------------------------------------------------------


Total Power
------------------------------------------------------------------------------
Total Leakage Power:       0.1879
------------------------------------------------------------------------------
Group                           Leakage       Percentage
                                Power         (%)
------------------------------------------------------------------------------
Sequential                       0.0127       6.756
Macro                            0.1556        82.8
IO                                    0           0
Combinational                   0.01962       10.44
------------------------------------------------------------------------------
Total                            0.1879         100
```

```
-------------------------------------------------------------------------
Rail                  Voltage   Leakage       Percentage
                                Power         (%)
-------------------------------------------------------------------------
VDDau                       0         0           0
VDDlu                    0.84   0.06477       34.47
VDDm                     1.08  0.003587       1.909
VDD                      1.08    0.1195       63.62

Clock                           Leakage       Percentage
                                Power         (%)
-------------------------------------------------------------------------
phy_rxclk_2                            0           0
phy_txclk_2                    0.0004786      0.2547
phy_rxclk_1                            0           0
phy_txclk_1                    3.39e-05     0.01804
clk                            0.001723      0.9171
-------------------------------------------------------------------------
Total                          0.002236        1.19
-------------------------------------------------------------------------



-------------------------------------------------------------------------
*       Power Distribution Summary:
*               Highest Average Power:
ethernet_mac_2/tx_fifo_module_from_dma/fifo1/dual_port_
   ram_4_tx_fifo/ram2P1024x32/ram (TSDN65LPA1024X32M8F):      0.02424
*               Highest Leakage Power:
ethernet_mac_2/tx_fifo_module_from_dma/fifo1/dual_port_
   ram_4_tx_fifo/ram2P1024x32/ram (TSDN65LPA1024X32M8F):      0.02424
*               Total Cap:      1.01625e-10 F
*               Total instances in design: 29235
*               Total instances in design with no power:  3596
*          Total instances in design with no activty:   115
*               Total Fillers and Decap:     0
-------------------------------------------------------------------------


Total leakage power = 0.187903 mW
 Cell usage statistics:
 Library tsdn65lpa128x16m8f_1v08, 1 cells ( 0.003900%) , 0.01016 mW (
   5.407063%  )
```

```
Library tsdn65lpa1024x32m8f_1v08, 6 cells ( 0.023402%), 0.145427 mW (
  77.394568%  )
Library tcbn65lp_LVL_c060217wc0d90d9 , 120 cells ( 0.468037%), 0.000175605 mW
  ( 0.093455%  )


Library unknown , 3881 cells ( 15.137096%) , 0.00630072 mW ( 3.353178%  )
Library tcbn65lpcgwc , 118 cells ( 0.460236%) , 0.000625092 mW ( 0.332667%  )
Library tcbn65lplvtwc , 3762 cells ( 14.672959%) , 0.0110529 mW ( 5.882233%  )
Library tcbn65lpwc , 17751 cells ( 69.234370%) , 0.0141619 mW ( 7.536835%  )
```

report_vcd_profile

```
report_vcd_profile
     -activity | -power | -internal | -switching
     [-step step]
     [-align_with_signal_edge signal_name @{rising | falling}]
     [-instances {inst_list}]
     [-exclude_instances {inst_list}]
     [-nworst integer]
     [-outfile filename]
     [-pg_net {rail_list}]
     [-propagate]
```

Creates VCD profile reports that are used to identify windows with maximum activity and power.

The command optionally reports toggles or power (on a per power net basis) spanning the entire VCD and queries activity/power for leaf-level instances. It also provides the option to align profiling with the rising or falling signal.

Optionally, you can run this command to create VCD profiles for switching, internal and leakage power.

Parameters

`-activity`
: Enables activity profiling. Reports the number of toggles for every window.

`-align_with_signal_edge signal_name @{rising | falling}`
: Starts profiling at the first rise or fall of the signal.

  Example: `-align_with_signal_edge clk1@rising`

`-exclude_instances {inst_list}`
: Disables counting of toggles or calculating power for the specified instances while reporting activity or calculating power for the entire design.

  Example: `-exclude_instances {inst2}`

`-instances {inst_list}`
: Enables profiling of the instances that you specify.

  Example: `-instances {inst1}`

`-internal` — Enables internal power profiling. Reports internal power in milliWatts (mW).

`-nworst` *`integer`* — Provides n top windows with maximum activity or power.

Default: 3

Example: `-nworst 10`

`-outfile` *`filename`*

Writes out the VCD profiling report into the file that you specify.

By default, the command writes out the report to `vcdprofile.report` file.

Example: `-outfile Activity.rep`

`-pg_net {rail_list}`

Performs power profiling for the specified power net.

Example: `-pg_net {vddhi}`

`-power` — Enables total power profiling. Reports total power in milliWatts (mW).

`-propagate` — Enables propogation when partial VCD is provided.

`-step step` — Specifies the step size, in nanoseconds (ns). Activity / Power profiling is carried out for every step size that you specify.

Example: `-step 15`

If you do not specify any step size, then the step size is automatically calculated as:

2 * (minimum clock period)

If you specify a VCD window using `read_activity_file`, for example, `read_activity_file -format VCD` *`file`* `-start` *`time`* `-end` *`time`*, then the step size is calculated as:

maximum {2 * minimum clock period, VCD window size / 50}

`-switching` — Enables switching power profiling. Reports switching power in milliWatts (mW).

Examples

Identifying VCD window with maximum activity

Read in Verilog netlist

Read in .libs

Read in SDC

```
read_activity_file -format VCD -vcd_scope top test.vcd
report_vcd_profile -activity -outfile activity.report -step 2
```

Results:

```
activity.report
The three intervals having the maximum activity:

 44to     46       80
 20to     22       60
 46to     48       50
```

Step(ns)		Instance	Activity
00to	02	Top	50
02to	04	Top	25
04to	06	Top	02
06to	08	Top	37.5
08to	10	Top	09
10to	12	Top	1.........

Identifying VCD window with maximum power

Read in Verilog netlist

Read in .libs

Read in SDC

```
read_activity_file -format VCD -vcd_scope top test.vcd
report_vcd_profile -power -outfile power.report -step 3.5
```

Results:

```
power.report
The three intervals having the maximum power:

45.5to   49.0     72.567
49.0to   52.5     63.550
52.5to   56.0     52.539
```

Step(ns)		Instance	Power(mW)
24.5to	28.0	Top	30.7
28.0	31.5	Top	44.6
70.0	73.5	Top	5.7
73.5	77.0	Top	6.3

reset_power_activity

```
reset_power_activity
```

Resets all activity in the database.

Parameters

None

set_dc_sources

```
set_dc_sources
     {netNameList}
     [-power | -ground]
     [-voltage value]
     [-global value]
     [-force]
```

Specifies the power and ground nets and sets the supply voltages. If you specify a voltage value, the value applies to either power or ground, but not both. The specified voltages can be used for both, static and dynamic power analysis.

Use this command after setting the name of the top module.

Parameters

`-force`	Creates power net if one doesn't exist and associates a voltage with it.
`-global` *`value`*	Sets the global power voltage value for signal integrity analysis. This option overrides the global power voltage value defined in the power domain or the -lib file. *Default:* The software uses the global power voltage value defined in the default power domain or the first .lib file, if the power domain is not specified.
`netNameList`	Specifies the list of nets to apply the supply information to. Always specify the *`netNameList`* parameter as the first argument in the command.
`-power | -ground`	Specifies whether the specified value is for nominal supply or ground voltage.
`-voltage` *`value`*	Specifies the value of the voltage. *Default:* `0.0` (units in volts) when you use the `-ground` option. With the `-power` option, the software uses the voltage from the default power domain or from the first .lib file if power domain is not specified.

Examples

- The following commands sets the nominal power with voltage value:

```
set_dc_sources -power VDD1 -voltage 1.2
set_dc_sources -power VDD2 -voltage 1.5
```

- The following command sets normal power without voltage value. The voltage is used from the first .lib file:

  ```
  set_dc_sources VDD
  ```

- The following command sets the ground supply:

  ```
  set_dc_sources -ground VSS
  ```

- The following command sets the global power voltage value to 1.2:

  ```
  set_dc_sources -global 1.2
  ```

set_default_switching_activity

```
set_default_switching_activity
     [-clock_gates factor]
     [-duty value]
     [-global_activity factor]
     [-hier hierarchy_name]
     [-input_activity factor]
     [-period value]
     [-seq_activity factor]
```

Specifies the switching activity for all primary inputs, nets, and other devices in the design whose activity has not been previously defined through user attributes, the toggle count format (TCF) file, the value change dump (VCD) file, or the tracing of the clock network.

Parameters

`-clock_gates` *`factor`*

Specifies the average number of times that a clock-gating cell switches in a clock cycle. Enter any positive number.

`-duty` *`value`*

Specifies the duty cycle, which is the probability that the signal is a logical 1.

Note: You must specify the `-period` parameter when assigning a duty.

Default: `0.5` (equivalent to a 50 percent duty cycle)

`-global_activity` *`factor`*

Specifies the average number of times that all unset nodes switch in a clock cycle. If not specified the fastest domain frequency will be used unless one turns off clock domains.

`-hier` *`hierarchy_name`*

A hierarchical name that the global activity is being assigned to. If not provided, the whole design is assumed.

`-input_activity` *`factor`*

Specifies the average number of times that a primary input switches in a clock cycle. It can be any positive number.

Default: `0.2` (Switches once every five clock cycles.)

`-period` *`value`*

Specifies the default operating period (1/frequency).

Specifies the period that the activity is referenced to. If not specified, the software uses the fastest clock in the design. If no clock is specified, the software generates an error and prompts you to enter one. Units in seconds (`s`), milliseconds (`ms`), microseconds (`us`), nanoseconds (`ns`), or picoseconds (`ps`).

Note: If you enter the period as a negative number, the software reverts back to using the default frequency.

Default: The time unit is defined in the first `.lib` file read during design import.

`-seq_activity` *`factor`*

Specifies the activity of outputs of sequential logic.

Activity Precedence

See "Activity Precedence" on page 219.

Examples

- The following command specifies that the primary inputs in the design will switch once every ten clock cycles, with a clock period of 7520 ns and a duty cycle of 50 percent:

```
set_default_switching_activity -input_activity 0.10 -period 7520ns -duty .5
```

- The following command specifies that all unswitched nodes in the design will switch once every ten clock cycles, with a clock period of 7520 ns and a duty cycle of 50 percent:

```
set_default_switching_activity -global_activity 0.10 -period 7520ns -duty .5
```

- The next example defines the switching activity, sequential activity, and the period:

```
set_default_switching_activity -input_activity 0.2 -period 10 \
    -seq_activity 0.1
```

- The next example sets the default global activity to 0.5 for the hierarchy `dma_dut`:

```
set_default_switching_activity -global_activity 0.5 -hier dma_dut
```

set_dynamic_power_simulation

```
set_dynamic_power_simulation
     [-help]
     [-period value]
     [-resolution value]
```

Specifies the period and resolution for a dynamic power simulation.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `set_dynamic_power_simulation` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man set_dynamic_power_simulation`.
`-period` *value*	The simulation period in nanoseconds (`ns`) that dynamic power analysis will use.
`-resolution` *value*	Specifies the transient time step size or resolution in picoseconds (`ps`).

Examples

- The following command sets up the period and resolution for a dynamic simulation:

  ```
  set_dynamic_power_simulation -period 5ns -resolution 100ps
  ```

set_power

```
set_power
     [-reset]
     [-leakage value]
     [-pg_net railname]
     [power | pwl]
     [-pwl]
     [-type { cell | instance } name ]
     [-sticky]
```

Specifies the amount of power for all or part of the design.

Parameters

`-leakage`	Specifies the target leakage power of the design, cell, instance, or power net. To specify leakage power for cells and instances you can use `-leakage` option in conjunction with `-type` [`cell` | `instance`] option. To specify leakage power of a certain power net in the design you can use `-pg_net` option along with `-leakage` option.
`-pg_net` *railname*	Optional. The specified power or pwl for the cell, instance or design is applied to the specified rail. When not specified, the power or pwl is applied to all the rails associated with the instance. This option is generally used for MSMV cells. When this option is not specified the behavior is different for pwl and power options. When not specified and power option is used, the power is applied for the whole design, cell, or instance. When not specified and pwl option is used, the pwl is applied to every rail of the cell or instance.
`-pwl`	Specifies that power argument (`power | pwl`) is a waveform rather than a power number.
`-reset`	This parameter causes all previous `set_power` commands to be ignored.
`-sticky`	Applies the simulation based PWL waveform as is, to the specified cell/instance for dynamic power analysis. Use this option only if you have specified `-pwl`.

`-type {cell | instance}` *`name`*

Specifies either a cell or instance name. If not specified, the power will be distributed to all rails.

[*`power`* | *`pwl`*]

Specifies power in milliwatts during static power calculation or piecewise linear (`-pwl`) current waveform in ns and mA, during dynamic power calculation for cell or instance. The command accepts a unit specification (e.g. 10w).

Examples

- The following command sets power for `vdd` rail to 10 milliwatts:

  ```
  set_power -pg_net vdd 10
  ```

- The following command sets a power value of `.005` milliwatts on power net `vdd` for instance `clkin_neg_L14_I1`:

  ```
  set_power -pgNet vdd -type instance Top/A/clkin_neg_L14_I1 .005
  ```

- The following command sets a power value of `.005` milliwatts for any cell named `INVX2` in the design:

  ```
  set_power -type cell INVX2 .005
  ```

- The following command defines a piecewise linear waveform of {0 0 10 0 10.1 5 10.5 0} for the cell of type `BUFX2MTL` for all power rails.

  ```
  set_power -pwl -pg_net {} -type cell BUFX2MTL {0 0 10 0 10.1 5 10.5 0}
  ```

- The following command specifies the leakage power for the cell `DFFRX1`.

  ```
  set_power -leakage -type cell DFFRX1 10mw
  ```

- The following commands specify the target leakage power for power nets `VDD1` and `VDD2`.

  ```
  set_power -leakage -pg_net VDD1 100mw
  set_power -leakage -pg_net VDD2 200mw
  ```

- The following command specifies the PWL description of current for the instance `SEHD130_1024X32X1CM8`. The `-sticky` option tells the software to use the specified PWL simulation based waveforms as is.

  ```
  set_Power -pwl -type instance SEHD130_1024X32X1CM8 -sticky {0 0 1 1 2 2 7 0}
  ```

set_power_analysis_mode

```
set_power_analysis_mode
     [-help]
     [-reset]
     [-analysis_view mmmc_view]
     [-corner {min|max}]
     [-create_binary_db {true | false}]
     [-handle_tri_state {false|true} ]
     [-honor_negative_energy {true | false}]
     [-ignore_control_signals {true | false}]
     [-ignore_inout_pin_cap {true | false}]
     [-leakage_scale_factor_for_temp scale]
     [-method { static | dynamic_vectorless | dynamic_vectorbased}]
     [-off_pg_nets net_list
     [-power_grid_library {library_list}]
     [-handle_glitch {true|false}]
     [-split_bus_power {true | false}]
     [-transition_time_method {min | avg | max}]
     [-write_static_currents {true | false}]
     [-x_transition_factor value]
     [-z_transition_factor value]
```

Sets the parameters for doing the power analysis.

Parameters

`-analysis_view mmmc_view`

Specifies the power analysis library view (for MMMC setup).

Note: You can also use `-view` instead of `-analysis_view` to specify the power analysis library view. Both the parameters are supported in this release.

If `report_power - view` *`view`* is specified, this view will override `-analysis_view` setting.

There is no default.

`-corner {min|max}`

Defines the library corner (for non-MMMC setup).

- `min` corner means power calculator will use min timing libraries for the power calculation.
- `max` corner means power calculator will use min timing libraries for the power calculation.

Default: `max`

`-create_binary_db {true | false}`

A value of `true` creates a binary database of power results. This file can be used in conjunction with the PS Viewer described in the *Power System Viewer Manual*. This is the recommended graphical user interface for debugging power information.

Default: `false`

`-handle_glitch {true | false}`

If set to `true`, and you run `report power`, the parameter reports glitch power as a separate component of total power and switching or internal power calculation does not include power due to glitches.

If set to `false`, glitch power will not be reported separately.

A transition is defined as a glitch when the time difference between two toggles are less than half of the (rise transition time + fall transition time).

Default: `false` By default, glitches are treated as normal transitions and contribute to the power reported.

Note: This parameter is currently not used during VCD profiling.

`-handle_tri_state {false|true}`

When set to `true`, all tristate device enable pin values will be taken into account when determining the propagation of the activity through tristate gates.

Default: `false`

`-help`

Outputs a brief description that includes type and default information for each `set_power_analysis_mode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man set_power_analysis_mode`.

`-honor_negative_energy {true | false}`

A value of `true` specifies that Encounter will keep negative internal energy numbers from the `.lib` internal power table. When set to false, negative power is treated as 0.

Default: `true`

`-ignore_control_signals {true | false}`

A value of `true` specifies that control signals will be ignored when propagating activity.

Default: `true`

`-ignore_inout_pin_cap {true | false}`

When set to `true`, ignores the bidirectional pin capacitances (`direction : inout`) defined for I/O cells in the .lib file, when calculating switching power; this also includes the *default* capacitance value for `inout` pins. However, this does not affect the internal power calculation.

Default: `false`

`-leakage_scale_factor_for_temp` *`scale`*

Performs a linear scaling of the leakage power for temperature in all libraries. This replaces the k-factor for temperature in the .lib file. For process and voltage, the scaling will continue to be based on the corresponding k-factors in the .lib file. A value of 0.8 scales the leakage power in the library to 80 percent. A value of 1.2 scales the leakage power in the library to 120 percent.

Default: `1`

`-method { static | dynamic_vectorless | dynamic_vectorbased}`

Specifies the type of analysis to be performed.

Default: `static`

`-off_pg_nets` *`net`*`_list` The name of the power rails that will be turned off. The net name can contain instance hierarchy if appropriate. PowerMeter will take this data into account when figuring out the list of power gates which are switched off.

Note: A net is turned off if all connected power gates are turned off by control signals. When a net is off, all of the instances connected to the net will only contribute leakage current into the connecting nets.

`-power_grid_library {`*`library_list`*`}`

Specifies the name of the Cadence power cell libraries (`.cl`)

`-reset` Resets all specified options back to default values. The reset option should be the first option specified, otherwise all options set prior to this will be reset.

`-split_bus_power {true | false}`

A value of `false` applies the internal power number to each individual bit.

A value of `true` divides the internal power number by bus width, before applying it to the bus bit.

Default: `false`

`-transition_time_method {min | avg | max}`

Specifies the minimum, maximum, or average transition time method that will be used with the integrated timer or the external TWF.

Default: `max`

`-write_static_currents {true | false}`

A value of `true` tells Encounter Power System to generate the current data files per net.

Default: `false`

`-x_transition_factor` *`value`*

When using VCD one can specify how transitions to and from X are counted. These transitions are defined as shown in Table 4-1 on page 249. Transitions to and from the X state will be treated as full transitions multiplied by the specified factor.

Note: 0 -> X -> 1 or 1 -> X -> 0 count as full transitions and they are not controlled by the parameter.

Default: `0.5`

`-z_transition_factor` *`value`*

When using VCD one can specify how transitions to and from Z are counted. These transitions are defined as shown in Table 4-1 on page 249. Transitions to and from the Z state will be treated as full transitions multiplied by the specified factor.

Note: 0 -> XZ-> 1 or 1 -> Z-> 0 count as full transitions and they are not controlled by the `parameter`.

Default: `0.25`

Table 4-1 Transitions from/to X and Z values and

Transition	Definition	Default
to/from X	[0 or 1 or Z] <-> X	0.5
to/from Z	[0 or 1] <-> Z	0.25

Examples

- The following command uses the negative internal power from the `.lib` files during analysis, specifies that the transitions that are to and from the Z state will be multiplied

by `.2`, and specifies that the transitions that are to and from the X state will be multiplied by `.45`,:

```
set_power_analysis_mode -honor_negative_energy true \
    -z_transition_factor .2 \
    -x_transition_factor .45
```

- The following command resets the `-honor_negative_energy` parameter to its default value:

```
set_power_analysis_mode -reset -honor_negative_energy
```

- The following command resets all `set_power_analysis_mode` parameters to their default values:

```
set_power_analysis_mode -reset
```

- The following command sets up a dynamic power analysis.

```
set_power_analysis_mode \
    -method dynamic_vectorless \
    -corner max \
    -create_binary_db true \
    -write_static_currents true \
    -honor_negative_energy true \
    -ignore_control_signals false \
    -power_grid_library { fast_allcells.cl }
```

- The following command sets up another dynamic power analysis.

```
set_power_analysis_mode \
    -method dynamic_vectorless \
    -analysis_view AVmac2off \
    -off_pg_nets VDDlu \
    -create_binary_db true \
    -write_static_currents true \
    -honor_negative_energy true \
    -ignore_control_signals false \
    -power_grid_library {accurate_stdcells.cl TSDN65LPA1024X32M8F.cl
```

set_power_calc_temperature

```
set_power_calc_temperature
     temp
```

Sets the temperature for static power calculation.

Parameters

temp

Specifies the temperature for static power calculation.

Note: The static power calculation software gets the temperature value of each instance from the following sources (in order of precedence):

- Instance temperature file
- `set_power_calc_temperature` command
- The current operating condition temperature

set_power_include_file

```
set_power_include_file
     file
```

Specifies a PowerMeter include file.

Note: The `set_power_include_file` command is used to pass PowerMeter options only in the dynamic mode (vectorbased and vectorless). If set_power_analysis_mode is set to `static`, the `set_power_include_file` command is ignored.

Parameters

`file`	Name of the include file. This is used when translating PowerMeter commands to Encounter Power System commands. This file will include all PowerMeter commands that can not be translated to Encounter PS commands.

Examples

- The following command specifies a power include file of `power.inc`:

  ```
  set_power_include_file power.inc
  ```

- The following example shows the contents of the `power.inc` file.

  ```
  # power.inc file must use the pre-defined design and power objects as "design"
  # and "ChipPwr" respectively. The power output object is not supported.
  design calCellsToIgnore BUFX1
  ChipPwr calRailVoltage VDD 1.2
  ```

set_power_output_dir

`set_power_output_dir` *`directory_name`*

Specifies the name of the directory that power information will be output to.

Parameters

`directory_name`	Specifies the name of the directory in which power analysis output information is written.

Examples

- The following command specifies the output directory called `static_power_AvallOn` in which the power information is written.

  ```
  set_power_output_dir static_power_AVallOn
  ```

set_powerup_analysis

```
set_powerup_analysis
     [-alwayson_net net]
     [-corners corner_list]
     [-default_stimulus pwl_list]
     [-macro_powerup_view [accurate | fast_accurate | fast] ]
     [-period value]
     [-pin_stimulus { pin1 pwl1 : pin2 pwl2 : ... }]
     [-resolution value]
     [-run_transient_analysis [ true | false ]]
     [-powerup_net net]
     [-save_simulation_data {voltages | currents | all}]
     [-spice_include file]
     [-spice_models file]
     [-spice_subckts file_list]
```

Specifies the necessary information needed for power-up analysis.

Parameters

`-alwayson_net` *net*`_list`

Specifies the always-on rail of a powergate.

`-corners` *corner*`_list`

Specifies the corners to be used from the SPICE model file.

`-default_stimulus` *pwl*`_list`

Specifies the power up piecewise linear stimulus to be used by Ultrasim in the transient simulation.

`-macro_powerup_view [accurate | fast_accurate | fast]`

Specifies the libgen view of the macro in macro-powerup flow.

Default: `fast`

`-period` *value* Specifies simulation period for Ultrasim simulation.

`-pin_stimulus {` *pin1 pwl1* `:` *pin2 pwl2* `}`

Specifies a piecewise-linear stimulus on pins.

This parameter is optional.

`-powerup_net` *`net`*`_list`

Specifies the nets that will be connected when the powergate is switched on.

`-resolution` *`value`* Specifies transient step size during Ultrasim simulation.

`-run_transient_ analysis [true | false]`

A value of `true` specifies that transient analysis is to be done.

`-save_simulation_data {voltages | currents | all}`

Saves the simulation data (UltraSim dynamic analysis data).

For example, the command `set_powerup_analysis -save_simulation_data currents`, saves only the current waveform flowing through each power gate cell.

`-spice_include` *`file`*

To include an external file to the SPICE dump that might contain custom Ultrasim commands.

`-spice_models` *`file`* Specifies the name of the SPICE models file. This SPICE information is needed for power-up analysis for powergates (power switches).

`-spice_subckts` *`file`*`_list`

Specifies an external SPICE file to use. You can provide one or more SPICE subcircuit files which would then be included in the SPICE deck.

Examples

- The following command sets up a power-up analysis:

```
set_powerup_analysis \
    -spice_models ../../DTMF_CHIP/SPICE_MODELS/CLN65LP_2d5_lk_v1d2.l \
    corners {TT_hvt TT_lvt TT TT_DIO DIO TT_18 TT_na} \
    -spice_subckts \
        {../../DTMF_CHIP/SPICE/cmos10lphvt_m.cdl \
        ../../DTMF_CHIP/SPICE/cmos10lphvt_m_pmk.cdl \
        ../../DTMF_CHIP/SPICE/cmos10lplvt_m.cdl \
        ../../DTMF_CHIP/SPICE/cmos10lplvt_m_pmk.cdl \
        ../../DTMF_CHIP/SPICE/cmos10lprvt_m.cdl \
        ../../DTMF_CHIP/SPICE/cmos10lprvt_m_pmk.cdl \
        ../../DTMF_CHIP/SPICE/ram_256x16A.cdl \
```

```
        ../../DTMF_CHIP/SPICE/ram_256x16A.spi \
        ../../DTMF_CHIP/SPICE/rom_512x16A.cdl \
        ../../DTMF_CHIP/SPICE/rom_512x16A.spi} \
    -powerup_net VDD_TDSPCore \
    -alwayson_net VDD_TDSPCore_R \
    -default_stimulus {0 0 100ps 0.8 500ps 0.8} \
    -run_transient_analysis false \
    -period 30ns \
    -resolution 100ps
```

- The following command sets up another power-up analysis:

```
set_powerup_analysis \
    -macro_powerup_view accurate \
    -spice_models ../DATA/vstorm/cln65lp_lk_v1d0.l \
    -corners {SS SS_LVT SS_HVT SS_25 SS_DIO SS_DIO_LVT SS_DIO_HVT} \
    -spice_subckts {tcbn65lp_121a.spi tcbn65lpcg_120a.spi \
        tcbn65lplvt_121a.spi tcbn65lplvtcg_120a.spi} \
    -powerup_net VDDlu \
    -alwayson_net VDDm \
    -default_stimulus {0 0 0.5ns 0 1ns 0.84} \
    -pin_stimulus {pcm_inst/pse[0] 0 0 0.5ns 0 0.9ns 0.84 : pcm_inst/pse[1] 0
0 0.5ns 0 0.9ns 0.84}
    -run_transient_analysis true \
    -period 30ns \
    -resolution 50ps
```

set_switching_activity

```
set_switching_activity
     [-reset]
     [-activity factor | -density transition_density ]
     [-clock clock_name]
     [-duty value]
     [-inst]
     [-net net_name | -port port_name | -pin pin_name ]
     [-period value]
     [-unclocked]
```

Specifies activity for nets, pins, and ports.

Parameters

`-activity` *`factor`*
: Specifies the activity for a net, pin, or top-level port. The *`activity_factor`* specifies the number of times the net, pin, or port switches in a clock cycle. Enter any positive number. If if the activity is set to 0.5, the activity happens once for every two clock cycles. If the activity is set to 2.0, the activity happens two times per clock cycle. There is no default value.

`-clock` *`clock_name`*
: Assigns global activity factor to all nets in domain of *`clock_name`* if and only if *`clock_name`* is the fastest clock in the domain set.

  To use this option, `-clock` and `-activity` must be used without any other options.

`-density` *`transition_density`*
: Specifies the transition density for a specific net, pin, or top-level port. Transition density specifies the number of transitions per second. it can be any positive number. There is no default value.

  Note: If you specify the `-density` parameter in conjunction with the `-period` parameter, the software issues a warning and the `-period` parameter is ignored.

`-duty` *`value`*	Specifies the duty cycle, which is the probability that the signal is a logical 1. The `-period` parameter must be specified to add a duty. If not specified, a default duty cycle is specified. *Default*: Uses the duty specified by the `set_default_switching_activity` command if the `-duty` parameter is not specified.
`-inst`	Specifies the global switching activity is for the hierarchical instance.
`-net` *`net_name`*	Specifies the name of the net. This parameter supports wildcards (`*`).
`-period` *`value`*	Specifies the period that the activity is referenced to. If not specified, a default frequency is used. If a negative number is specified, the static power calculation engine reverts back to the default frequency. Units in seconds (`s`), milliseconds (`ms`), microseconds (`us`), nanoseconds (`ns`), or picoseconds (`ps`). **Note:** If you specify the `-period` parameter in conjunction with the `-density` parameter, the software issues a warning and the `-period` parameter is ignored. *Default*: The time unit is defined in the first `.lib` file read during design import. If the time unit is not defined, the period specified in the `set_default_switching_activity` command is used.
`-pin` *`pin_name`*	Specifies the name of a pin. The pin names are in the following format: *`instance-name + default-hierarchy-separator + pin`*. This parameter supports wildcards (`*`)
`-port` *`port_name`*	Specifies the name of a top-level port. The port names are in the following format: *`top-level-port`*. This parameter supports wildcards (`*`)
`-reset`	Resets all specified options back to default values. The reset option should be the first option specified, otherwise all options set prior to this will be reset.
`-unclocked`	Applies *`activity`* (with reference *`period`*) to all nets that have no clock domain (`-unclocked`). To use this option, `-unclocked` must be used with `-activity` and `-period` with no other options.

Activity Precedence

See "Activity Precedence" on page 219.

Examples

- The following command specifies that net `n189` will switch once every ten clock cycles, with a clock period of `7520ns` and a duty cycle of 50 percent:

  ```
  set_switching_activity -net n189 -activity 0.10 -period 7520ns -duty .5
  ```

- The following command sets the transition density of port `out_cpu_top` to `20000` transitions per second:

  ```
  set_switching_activity -port out_cpu_top -density 20000
  ```

set_timing_window_file

```
set_timing_window_file
     [-reset
     [-scope scope]
     twf_file
```

Specifies the external Timing Window File (TWF) to read during static or dynamic power calculation. This file will be used to get clocks, constants, external load, slews, timing windows, and slack information. The external timing window file specification is optional. In absence of the TWF, this information is automatically derived using Common Timing Engine for power calculation.

Important

Command `set_timing_window_file` is now obsolete and will be removed in the next major release of the software. This command has been replaced by `read_twf` command.

Parameters

`-reset`	Clears a previously read TWF file.
`-scope scope`	Specifies the scope for the given hierarchical timing window file.
`twf_file`	The name of the timing window file.

Examples

- The following command specifies a timing window file of `dma_mac.twf`

```
set_timing_window_file dma_mac.twf
```

write_tcf

```
write_tcf tcf_file_name
     [-primary_input]
     [-seq_output tcl_filename]
```

Writes out the propagated switching activity information to a toggle count format (TCF) file. The TCF file contains switching activity information, toggle count information, and the probability of the net or pin being in the logic1 state.

Parameters

`-primary_input`	Generates a TCF file that includes only the primary inputs in the design.
`-seq_output tcl_filename`	Includes all the activity for the sequential outputs in the design.
`tcf_file_name`	Specifies the name of the output TCF file.

Example

- The following command writes the propagated switching activity information to TCF file `mmm.tcf`:

  ```
  write_tcf mmm.tcf
  ```

5

Delay Calculation Commands

cleanupExcludeNet

```
cleanupExcludeNet
```

Clears excluded net settings that were specified with the `setExcludeNet` command. Any nets that were previously excluded from delay calculation using the `setExcludeNet` command are now included in delay calculation. You do not have to rebuild the timing graph to include the previously-excluded nets. Use this command after specifying the nets that are to be excluded from delay calculation.

Parameters

None

delayCal

```
delayCal
     [-nocell [-net fileName]]
     [-idealclock]
     [-sdf fileName]
     [-nosetdrv]
     [-lumpcap]
     [-version [2.1 | 3.0]]
     [-floatpin]
     [-mergeRecoveryRemoval]
     [-mergeSetupHold]
     [-splitIOPath]
     [-splitUnateDelay]
```

Calculates the delays, and outputs the results in Standard Delay Format (SDF) format. The delays are calculated for interconnect, or for both interconnect and cells. Delays are calculated using the algorithm specified with the setDelayCalMode command. If you do not first specify an algorithm, the Encounter delay calculation engine is used by default. Use this command after setting the default delay values and running extract RC.

Note: ILMs must be in a flattened state before you run this command. To flatten ILMs, run the `flattenIlm` command first.

The Encounter software uses default delay values for nets that exceed 1,000 terminals. To change the net terminal limit, use the `setUseDefaultDelayLimit` command.

Net default delay values are:

- Net delay: 1 ns
- Transition time: 0 ps
- Net load: 0.5 pF

To change the default delay values, use the following commands:

- `setDefaultNetDelay`
- `setInputTransitionDelay`
- `setDefaultNetLoad`

Delays are calculated differently, depending on the number of terminals (fanouts) a net has and the Elmore time constant. The following table lists the calculation modes and related controls used for delay calculation.

	Fanouts >= 1,000	**1,000 > Fanouts >= 100**	**100 > Fanouts and Elmore > 5ps**	**5ps > Elmore**
Algorithm	Uses the default delay parameters.	Uses simplified delay calculation mode.	Uses full RC delay calculation mode.	Uses simplified delay calculation mode.
Cell Delay	Uses lumped C lookup with a default value of 0.5 pF. The value is controlled by the `setDefaultNetLoad` command.	Uses lumped C lookup from total parasitics on the net.	Uses full RC.	Uses lumped C lookup from total parasitics on the net.
Wire Delay	Uses the default value of 1 ns. The value is controlled by the `setDefaultNetDelay` command.	Uses Elmore delay.	Uses full RC.	Uses Elmore delay.
Driver Slew Rates	Uses the default value of 0 ps. The value is controlled by the `setInputTransitionDelay` command.	Uses lumped C lookup from total parasitics on the net.	Uses full RC.	Uses lumped C lookup from total parasitics on the net.
Interconnect Slew Degradation	None	None	Uses full RC. (Slews are degraded across interconnect.)	None
Controls	Fanout threshold is controlled by `setUseDefaultDelayLimit` command. The default value is 1,000 fanouts.	Fanout threshold is controlled by the `-elmore` option in the `buildTimingGraph` command. The default value is 100 fanouts.		No control for minimum Elmore threshold.

Parameters

`-floatpin` Includes the `IOPATH` statements with floating terminals. The `IOPATH` statements correspond to the timing arcs in the timing library.

`-idealclock`	Forces the clock nets on each flip-flop input pin to use 0 ps. Use the setInputTransitionDelay command to change the default value of the clock.
`-lumpcap`	Uses lump capacitance for delay calculation.
`-mergeRecoveryRemoval`	Merges the `RECOVERY` and `REMOVAL` checks into `RECREM` checks in the SDF file. You can use this parameter to avoid negative hold times in the SDF file. *Default*: Writes out separate `RECOVERY` and `REMOVAL` checks in the SDF file.
`-mergeSetupHold`	Merges the `SETUP` and `HOLD` checks into `SETUPHOLD` checks in the SDF file. You can use this parameter to set a negative value for the hold time in the SDF file. *Default*: Writes out separate `SETUP` and `HOLD` checks in the SDF file.
`-net` *`fileName`*	Calculates delays only for the nets in the specified input file. *Default:* Calculates delays for all nets
`-nocell`	Calculates delays for interconnects only, and reads the cell timing from the timing library. *Default:* Calculates delays for both cells and interconnects
`-nosetdrv`	Ignores the `set_drive` constructs in the timing constraints file.
`-sdf` *`fileName`*	Writes out the delay calculation results to the specified SDF file. **Note**: When the software is in multi-mode multi-corner analysis mode, you should not use this parameter to generate the SDF file. Use the `write_sdf` command for this purpose.

`-splitIOPath`	Writes out the `posedge` and `negedge` statements for each `IOPATH` separately in the SDF file.
	
	For example, if you specify `-splitIOPath`, the following `IOPATH` statement:
	
	`(IOPATH  H01 N01 (58:90:144) (62:96:154))`
	
	is instead written out as:
	
	`(IOPATH (posedge H01) N01 (58:90:144) ())`
	`(IOPATH (negedge H01) N01 () (62:96:154))`

`-splitUnateDelay`	Writes the positive edge delays (rise -> rise and rise -> fall) and the negative edge delays (fall -> rise and fall -> fall) separately in the SDF file when there is non-unate timing between two pins. Default: Merges the rise -> rise and fall -> rise edge delays into rise/fall -> rise, and merges the rise -> fall and fall -> fall edge delays into rise/fall -> fall for non-unate timing between two pins.
`-version`	Specifies which version of the SDF file to use when writing out the delay calculation results. *Default:* `3.0`

Example

- The following command calculates both interconnect and cell delays, and writes them to the `TOPCHIP_SP.sdf` file when the rise and fall times of clocks are set to 0 ps:

  ```
  delayCal -idealclock -sdf TOPCHIP_SP.sdf
  ```

getDelayCalMode

```
getDelayCalMode
     [-ecsmType]
     [-engine]
     [-honorSlewPropConstraint]
     [-reportOutBound]
     [-signalStormUseArcPCap]
     [-signalStormUseMultiPCap]
     [-quiet]
     [-signoff]
     [-signalStormMixIoPathAtMulti]
```

Displays the following information about a setDelayCalMode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setDelayCalMode` parameters.

Parameters

`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `setDelayCalMode` for descriptions of the parameters you can specify.
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setDelayCalMode` parameters in Tcl list format.

Example

- The following command displays the current setting of the `-ecsmType` parameter:

```
getDelayCalMode -ecsmType
```

The software displays the following information:

```
-ecsmType ecsmOnDemand                  # enums={ecsmAllCells ecsmOnDemand},
                                          default=ecsmOnDemand

ecsmOnDemand
```

- The following command displays the current settings for all of the `setDelayCalMode` parameters:

  ```
  getDelayCalMode
  ```

  The software returns the following information:

  ```
  -ecsmType ecsmOnDemand                  # enums={ecsmAllCells ecsmOnDemand},
  default=ecsmOnDemand
  -engine feDC                            # enums={feDC signalStorm default},
  default=default, user setting
  -honorSlewPropConstraint true           # bool, default=true
  -reportOutBound false                   # bool, default=false
  -signalStormUseArcPCap false            # bool, default=false
  -signalStormUseMultiPCap false          # bool, default=false
  -signOff false                          # bool, default=false


  {ecsmType ecsmOnDemand} {engine feDC} {honorSlewPropConstraint true}
  {reportOutBound falsse} {signalStormUseArcPCap false} {signalStormUseMultiPCap
  false} {signOff false}
  ```

- The following command displays the current settings for all `setDelayCalMode` parameters in Tcl list format only:

  ```
  getDelayCalMode -quiet
  ```

  The software returns the following information:

  ```
  {ecsmType ecsmOnDemand} {engine feDC} {honorSlewPropConstraint true}
  {reportOutBound falsse} {signalStormUseArcPCap false} {signalStormUseMultiPCap
  false} {signOff false}
  ```

reportDelayCalculation

```
reportDelayCalculation
     -from port_or_pin_name
     -to port_or_pin_name
     [-delaytable]
     [-outfile filename ]
     [-min]
     [-max]
     [-thresholds]
     [-view viewname]
```

Reports the delay calculation information for a cell or net timing arc.

Important

This command reports the base delay values only. It does not report the incremental delay values.

Parameters

`-delaytable` Retrieves the input slew and output capacitance values from the library lookup table that were interpolated by the delay calculation engine to calculate the delay values. This parameter is useful if you want to check how the delay values printed at the report were calculated.

`-from` *`port_or_pin_name`*

Specifies the name of the `From` port or pin. If the From port or pin is on a net, the Delay type field in the report shows `net`.

`-max` Indicates that the delay calculation report is for the worst-case operating conditions in min-max mode. If neither `min` nor `max` arguments are specified, then the default mode is set to `max`.

`-min` Indicates that the delay calculation report is for the best-case operating conditions in min-max mode. If neither `min` nor `max` arguments are specified, then the default mode is set to `max`.

`-outfile` *`filename`*

Specifies the name of the output file in which to save the delay calculation report.

`-thresholds` Displays the rise delay, rise slew, fall delay, fall slew, and slew delay threshold values for the From and To pins.

`-to` *`port_or_pin_name`*

Specifies the name of the To port or pin. If the To port or pin is on a net, the Delay type field in the report shows `net`.

`-view` *`viewname`*

Specifies the view name for which to print the delay calculation report when using the multi-mode multi-corner (MMMC) mode.

Example

The following command shows the delay calculation report for the From pin `i_2/Y` and To pin `i_9/CK`:

```
reportDelayCalculation -from i_2/Y -to i_9/CK
#################################################################
# Generated by:          Cadence First Encounter 08.10-e052_1
# OS:                    Linux x86_64(Host ID amd64pv75)
# Generated on:          Mon Jun 9       11:13:26
# Command:               reportDelayCalculation -from i_2/Y -to i_9/CK
                          -outfile delaycalc.rpt
#################################################################

From pin  : i_2/Y
To pin    : i_9/CK
Delay type: net

               Rise   Rise     Fall    Fall      Slew
Thresholds:    Delay  Slew     Delay   Slew     Derate
From pin  :    50     10->90    50     90->10   1.000
To pin    :    50     10->90    50     90->10   1.000

-------------------------------------------------------------
RC Summary for net n_6
-------------------------------------------------------------
Number of capacitance : 11
Net capacitance       : 0.004460 pF
Total capacitance     : 0.010284 pF
Number of resistance  : 10
Total resistance      : 50.354072 Ohm
-------------------------------------------------------------
                             Rise          Fall
-------------------------------------------------------------
```

```
Net delay                :0.000200 ns    0.000200 ns
From pin transition time :0.166700 ns    0.105900 ns
To pin transition time   :0.166700 ns    0.105900 ns
-------------------------------------------------------------
```

saveExcludeNet

```
saveExcludeNet -file fileName
```

Reports the list of nets that are excluded from delay calculation. This includes the nets excluded by the setExcludeNet command and the nets that exceed the default value of 1,000 pins. Use this command after performing delay calculation.

Parameters

`-file` *`fileName`*	Specifies the name of the excluded net file.

Example

- The following command saves the list of excluded nets to a file named `myfile.exc`:

  ```
  saveExcludeNet -file myfile.exc
  ```

saveSignalStormConstraint

```
saveSignalStormConstraint -outfile fileName
```

Translates constraint information from Encounter, such as boundary slews and loads, into SignalStorm constraint syntax, and outputs it into a specified file. This file can then be included in, or sourced by, a SignalStorm script to run standalone SignalStorm. Use this command after loading the netlist and timing constraints.

Parameters

`-outfile fileName` Specifies the file to which to output the constraint information.

setDefaultNetDelay

`setDefaultNetDelay` *`delay`*

Sets the default net delay for delay calculation. The Encounter software uses this default value for nets that exceed 1,000 terminals. The delay for nets with fewer than 1,000 terminals is calculated using the delayCal command.

Parameters

`delay`	Specifies the default net delay. *Default:* 1ns

Example

- The following command sets the net delay to 3 ns for nets that exceed 1,000 terminals:

  ```
  setDefaultNetDelay 3ns
  ```

setDefaultNetLoad

```
setDefaultNetLoad load
```

Sets the default net load for delay calculation. The Encounter software uses this default value for nets that exceed 1,000 terminals.

Parameters

load	Specifies the default net load. *Default:* `0.5pF`

setDelayCalMode

```
setDelayCalMode
     [-help]
     [-reset]
     [-ecsmType {ecsmAllCells | ecsmOnDemand}]
     [-engine {feDc | signalStorm | default}]
     [-signalStormUseArcPCap {true | false}]
     [-signalStormUseMultiPCap {true | false}]
     [-honorSlewPropConstraint {true | false}]
     [-signoff {true | false}]
     [-reportOutBound {true | false}]
     [-signalStormMixIoPathAtMulti {true | false}]
```

Sets global parameters for delay calculation. Parameters you specify with the `setDelayCalMode` command are then used automatically whenever you run delay calculation, either directly through the `delayCal` command, or internally during timing analysis and in-place optimization. The `setDelayCalMode` parameters affect the behavior of the following command:

- `optDesign`

Use the getDelayCalMode on page 269 to return the current settings for the `setDelayCalMode` command.

You can also use the `setDelayCalMode` command during the run to switch processes, similar to using `setOpCond` to control `.lib` process corner selection.

Parameters

`-ecsmType {ecsmAllCells | ecsmOnDemand}`

Determines how library cells are translated.

Default: `ecsmOnDemand`

Note: These parameters can only be used if you also specify the `-engine signalStorm`.

`ecsmOnDemand`

Incrementally translates library cells. Encounter initially translates only the cells that are used in the design. If other cells are needed (for example, during in-place optimization), Encounter translates the necessary cells.

Note: This parameter cannot be used in conjunction with the `ecsmAllCells` parameter.

`ecsmAllCells`

Translates all library cells at the same time.

Note: This parameter cannot be used in conjunction with the `ecsmOnDemand` parameter.

`-engine {feDc | signalStorm | default}`

Specifies which delay calculation engine to use for delay calculation.

Default: `default`

`feDc`	Uses the Encounter algorithm for delay calculation. **Note**: This parameter cannot be used in conjunction with the `signalStorm`, `ecsmOnDemand`, or `ecsmAllCells` parameters.
`signalStorm`	Uses the SignalStorm algorithm for delay calculation.
`default`	Uses the default delay calculation engine, which is based on the Encounter algorithm. Setting the `default` value is equivalent to setting the `feDc` value.

`-help`

Outputs a brief description that includes type and default information for each `setDelayCalMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setDelayCalMode`.

`-honorSlewPropConstraint {true | false}`

Determines whether to propagate the slew from the disabled timing arcs to the output pin. When this parameter is set to `false`, the slew from the disabled timing arc affects the delay of the next stage. Setting this parameter to `false` results in more accurate slew computation for the disabled timing arcs but can take longer run time.

When set to `true`, the timing constraints defined using commands, such as `set_disable_timing` and `set_case_analysis` impact the slew propagation, and the slew from the disabled timing arcs is not propagated.

Default: `true`

`-reportOutBound {true | false}`

Generates a report that contains a list of index values (input transition and output loading) in the delay tables that are beyond the index range. The index values detected by this parameter can cause inaccurate delay values to be calculated. The report generated by this parameter can be used for debugging.

Default: `false`

`-signalStormMixIoPathAtMulti {true | false}`

Determines whether to check the cell delay of all parallel drivers of the net to find the minimum and maximum delay values, and then assign those values back to all the delays for delay calculation.

When set to `true`, the software checks the cell delays and assigns the minimum and maximum values. When set to `false`, the software does not check the cell delays.

Note: This parameter can only be used with the `signalStorm` delay calculation engine.

`-signalStormUseArcPCap {true | false}`

Specifies whether to use the arc-based ECSM pin capacitance model for performing delay calculation.

Set this parameter to `true` for using arc-based ECSM pin capacitance.

Note: You must enable the `soceSignalStormUseNextStageLoad` variable before using this parameter. This variable determines whether to use the next stage load while performing delay calculation. To enable this variable, use the following command:

```
setVar soceSignalStormUseNextStageLoad 1
```

`-signalStormUseMultiPCap {true | false}`

Specifies whether to use the multi-piece ECSM pin capacitance model, which accounts for Miller capacitance while performing delay calculation.

Set this parameter to `true` for using multi-piece ECSM pin capacitance.

Note: You must enable the `soceSignalStormUseNextStageLoad` variable before using this parameter. This variable determines whether to use the next stage load while performing delay calculation. To enable this variable, use the following command:

```
setVar soceSignalStormUseNextStageLoad 1
```

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setDelayCalMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-signoff {true | false}`

Enables sign-off quality delay calculation mode. In this mode, Encounter configures internal delay calculation parameters to provide the highest accuracy. This mode also provides consistency with ETS and CeltIC NDC.

Default: `false`

Note: By default, the SOCE implementation flow setting for the delay calculator is biased towards better performance at the expense of some accuracy.

Example

- The following command sets SignalStorm as the engine to use for delay calculation, and translates only the library cells used in the design:

  ```
  setDelayCalMode -engine signalStorm -ecsmType ecsmOnDemand
  ```

- The following command resets the `-engine` parameter to its default value:

  ```
  setDelayCalMode -reset -engine
  ```

- The following command resets all `setDelayCalMode` parameters to their default values:

  ```
  setDelayCalMode -reset
  ```

Related Topics

- Signoff in the *Encounter Flat Implementation Flow Guide*

setExcludeNet

```
setExcludeNet {-file filename | -template}
```

Specifies an exclude net file to use during delay calculation. Normally, nets that exceed the default value of 1,000 pins are excluded from delay calculation, and default delay values are used. If you specify an exclude net file, nets (including hierarchical nets) listed in the file are excluded from calculation, and their corresponding delay values specified in the file are used. Nets usually specified in the file include high-fanout nets that increase the run time of the program.

The exclude net file can have one of two formats:

- *netName netDelay netOutputTran netLoad*
- *netName netDelay*

Note: Although a net is considered as ideal by the timing analysis engine, it might be buffered in order to reduce the fanout count or the total pin load. Therefore, if you want the ideal net to remain unchanged, you will need to apply the `set_dont_touch` attribute to it.

Parameters

`-file` *filename*	Specifies the name of the exclude net file.
`-template`	Generates a sample file that contains the net and delay formats to use to specify the file.

Examples

- The following command sets an exclude net file named `myfile.exc`:

  ```
  setExcludeNet -file myfile.exc
  ```

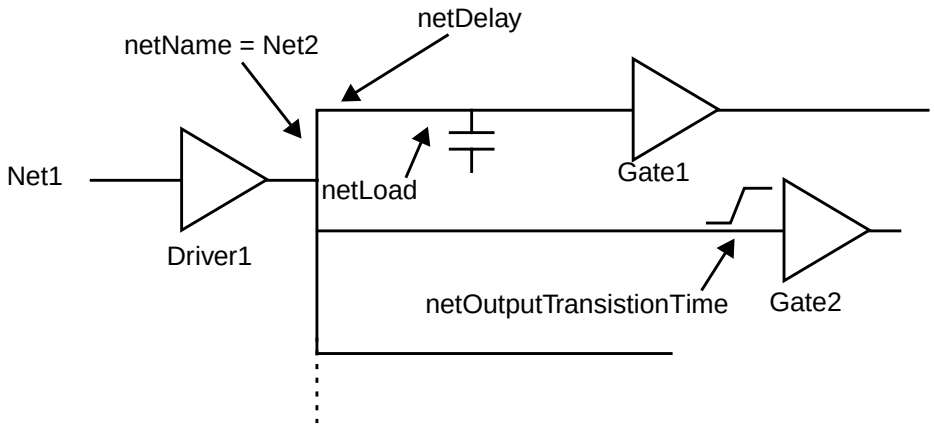
  Each of the excluded nets use the corresponding delay value specified in the `myfile.exc` file.

- The following command creates a sample file that contains net and delay formats to use to create an exclude file:

  ```
  setExcludeNet -template
  ```

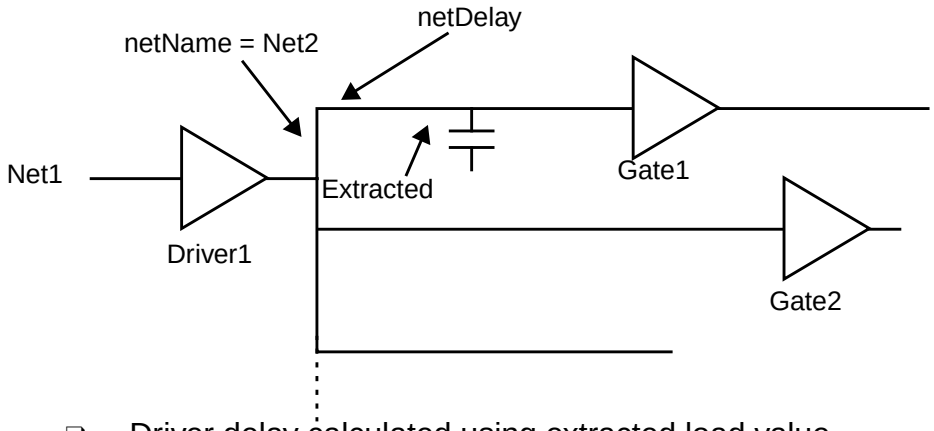
The following figures show the usage of the exclude net file format file:

Figure 5-1 Format 1: <netName> <netDelay> <netOutputTran> <netLoad>



- Driver delay calculated using `netLoad` value.
- Gate delay calculated using `netOutputTran` value.
- Default delay values are used for remaining nets with a terminal count greater or equal to 1000.

Figure 5-2 Format 2: <netName> <netDelay>



- Driver delay calculated using extracted load value.
- Gate delay calculated using extracted load value.
- Default delay values are used for remaining nets with a terminal count greater or equal to 1000.

The timing units for the file are:

Symbol	Unit
fs, Fs	10^{-15} Second
ps, Ps	10^{-12} Second
ns, Ns	10^{-9} Second
us, Us	10^{-6} Second
ms	10^{-3} Second

setInputTransitionDelay

`setInputTransitionDelay` *`delay`*

Sets the default input transition delays and the ideal clock transition time for delay calculation. The Encounter software uses this default for nets that exceed 1,000 terminals.

In addition, the default input transition value is applied to those pins to which the slew was not propagated. For example:

- Tiehi or tielo pins
- Macro output pins, which do not have a related input pin

  Note: A macro output pin, which has a related input pin will most likely have slew propagation from the related input pin to the output pin.
- Macro output pins for which all the timing arcs are disabled

Note: You must have a design loaded before you can use this command.

Parameters

`delay`	Specifies the input transition delay. *Default:* `0ps`

Example

- The following command sets the input transition delay to 300ps for nets that exceed 1,000 terminals:

  ```
  setInputTransitionDelay 300ps
  ```

setInstTemperature

```
setInstTemperature [-min | -max] [-early | -late] [-list] [-reset]
     [-defaultTemperature temperature_in_c] [-infile { fileNames }]
```

Loads the instance-based temperature information that is output by power-rail analysis tools, such as VoltageStorm PE, to account for the effects of temperature when calculating the delays and slews. You can read in four different files for early-max, early-min, late-max, and late-min temperature analysis.

Note: This command is supported only when using the SignalStorm delay calculator.

Parameters

`-defaultTemperature` *`temperature_in_c`*

Specifies the default temperature for the whole chip. Units in celsius.

Note: The `DEFAULT TEMPERATURE` statement in the temperature file takes precedence over this parameter.

`-early | -late`

Applies the effects of temperature to the early path (data hold/clock setup) or late path (data setup/clock hold) in your design. Use the `-early` or `-late` parameter in conjunction with the `-min` or `-max` parameter to account for the effects of early-max, early-min, late-max, or late-min temperature analysis.

`-infile {`*`fileNames`*`}`

Specifies the names of the temperature files that contain the instance-based temperatures. When used with the `-defaultTemperature` parameter, the instances that do not appear in these files will use the specified default temperature. The format of the temperature file is similar to the IR-drop file format. For example:

```
DEFAULT TEMPERATURE 25.3
- DTMF_INST/RESULTS_CONV_INST/i_78080/i_4456 75.2 ;
END ;
```

`-list`

Provides a list of all specified temperature files.

`-min | -max`	Applies the effects of temperature to the minimum operating condition or maximum operating condition. Use the `-min` or `-max` parameter in conjunction with the `-early` or `-late` parameter to account for the effects of early-max, early-min, late-max, or late-min temperature analysis. **Note**: If the `-min` or `-max` parameter is used without the `-early` or `-late` parameter, the temperature is applied to both early and late analysis. To apply the temperature for late analysis only, use the `-min` or `-max` parameter with the `-late` parameter. To apply the temperature for early analysis only, use the `-min` or `-max` parameter with the `-early` parameter.
`-reset`	Resets the temperature files.

Examples

- The following commands apply the instance-based temperatures for early-min, late-min, early-max, and late-max temperature analysis:

```
setInstTemperature -early -min -infile { earlymin.temp }
setInstTemperature -late -min -infile { latemin.temp }
setInstTemperature -early -max -infile { earlymax.temp }
setInstTemperature -late -max -infile { latemax.temp }
```

- The following commands apply the instance-based temperatures for late-min and late-max temperature analysis:

```
setInstTemperature -late -min -infile { latemin.temp }
setInstTemperature -late -max -infile { latemax.temp }
```

- The following commands apply the instance-based temperatures for early and late temperature analysis:

```
setInstTemperature -early -inflie { early.temp }
setInstTemperature -late -infile { late.temp }
```

setIrDropInstVoltage

```
setIrDropInstVoltage [-min | -max] [-early | -late] -infile { fileName } -list
     -reset
```

Loads the instance-based IR-drop information that is output by power-rail analysis tools, such as VoltageStorm PE, to account for the effects of IR drop when calculating the delays and slews. You can read in four different files for early-max, early-min, late-max, and late-min IR-drop analysis.

Note: This command is supported only when using the SignalStorm delay calculator.

Important

You cannot use the `setIrDropVoltage` command when the software is in multi-mode multi-corner timing analysis mode. Instead, use the `create delay corner` and `update delay corner` commands to specify IR drop information.

Parameters

`-early | -late`	Applies the effects of IR-drop to the early path (data hold/clock setup) or late path (data setup/clock hold) in your design. Use the `-early` or `-late` parameter in conjunction with the `-min` or `-max` parameter to account for the effects of early-max, early-min, late-max, or late-min IR-drop analysis.
`-infile {`*`fileNames`*`}`	Specifies the names of the IR-drop files to load.
`-list`	Provides a list of all specified IR-drop files.
`-min | -max`	Applies the effects of IR-drop to the minimum operating condition or maximum operating condition. Use the `-min` or `-max` parameter in conjunction with the `-early` or `-late` parameter to account for the effects of early-max, early-min, late-max, or late-min IR-drop analysis. **Note**: If the `-min` or `-max` parameter is used without the `-early` or `-late` parameter, the IR drop is applied to both early and late analysis. To apply the IR drop for late analysis only, use the `-min` or `-max` parameter with the `-late` parameter. To apply the IR drop for early analysis only, use the `-min` or `-max` parameter with the `-early` parameter.
`-reset`	Resets all IR-drop files.

Examples

- The following commands apply the IR drop for early-min, late-min, early-max, and late-max IR-drop analysis:

```
setIrDropInstVoltage -early -min -infile { earlymin.ir }
setIrDropInstVoltage -late -min -infile { latemin.ir }
setIrDropInstVoltage -early -max -infile { earlymax.ir }
setIrDropInstVoltage -late -max -infile { latemax.ir }
```

- The following commands apply the IR drop for late-min and late-max IR-drop analysis:

```
setIrDropInstVoltage -late -min -infile { latemin.ir }
setIrDropInstVoltage -late -max -infile { latemax.ir }
```

- The following commands apply the IR drop to early and late IR-drop analysis:

```
setIrDropInstVoltage -early -inflie { early.ir }
setIrDropInstVoltage -late -infile { late.ir }
```

setUseDefaultDelayLimit

`setUseDefaultDelayLimit` *`nrPins`*

Sets the pin number threshold for nets for delay calculation. Nets that have more than the specified number of pins are excluded from delay calculation, and use the default delay values instead.

Net default delay values are:

- Net delay: 1 ns
- Transition time: 0 ps
- Net load: 0.5 pF

To change the default delay values, use the following commands:

- `setDefaultNetDelay`
- `setInputTransitionDelay`
- `setDefaultNetLoad`

Parameters

`nrPins`	Specifies the pin threshold number. *Default:* `1,000`

Example

- The following command sets the pin threshold number for nets to 2,000:

  ```
  setUseDefaultDelayLimit 2000
  ```

  Nets that exceed pin count of 2,000 are excluded from delay calculation and use the default delay values.

setUseElmoreDelayLimit

`setUseElmoreDelayLimit` *`nrPins`*

Sets the lower limit for the number of pins per net to use Elmore delay. The software does not use Elmore delay for nets that have pins lesser than the number specified with this command.

By default, the software uses a lower limit of 100 pins per net. This means that any net that has more than or equal to 100 pins will use Elmore delay.

Tip

To set the upper limit for the number of pins for Elmore delay calculation, use the `setUseDefaultDelayLimit` command.

For nets that contain lesser number of pins than specified with this command, the software performs full RC delay calculation.

Parameters

`nrPins`	Specifies the lower limit for the number of pins per net to use Elmore delay. *Default:* `100`

Example

- The following command sets the pin threshold number for nets to 200:

  ```
  setUseElmoreDelayLimit 200
  ```

  Nets that have a pin count lesser than 200 are excluded from Elmore delay calculation, and the software uses full RC delay calculation for such nets.

setWireDelayFactor

`setWireDelayFactor` *`number`*

Specifies the factor by which to scale interconnect delays (wire delays). Use this command before performing timing analysis or delay calculation.

Parameters

`number`	Specifies the factor by which to scale the wire delays.

translateSNDCSetupFile

```
translateSNDCSetupFile
     sndcSetupFile
     outputFile
```

Uses a SignalStorm delay calculator setup file as input and converts the transition time and capacitative loading portions to timing constraints commands, such as `set_annotated_transition` and `set_load`. The timing constraints commands are included in the specified output file.

Parameters

sndcSetupFile	Specifies the name of the SignalStorm delay calculator setup file that will be converted.
outputFile	Specifies the name of the output file, which contain the timing constraints commands.

Example

The following command uses a SignalStorm delay calculator setup file named `sndc.setup` and generates the `constraints.rpt` file, which contains the timing constraints commands:

```
translateSNDCSetupFile sndc.setup constraints.rpt
```

writeSetLoad

```
writeSetLoad
     [-direct]
     [-includePinCap]
     -file outFile
```

Writes the capacitive loading on each net to an output file. By default, the capacitance value does not include the pin capacitance. You can either set the load using the `set_load` command or the software uses the load defined in the SPEF file.

Note: The command is currently not supported in multi-mode multi-corner mode.

Parameters

`-direct`	Refers the ports directly for which the capacitative loading will be written. This option removes the dependency of using the `get_nets` command.
`-file outFile`	Specifies the name of the output file.
`-includePinCap`	Includes the pin capacitance while writing out the capacitive loading on each net to the output file.

Examples

- The following command writes the capacitive load in the report file `load.rpt`:

  ```
  writeSetLoad -file load.rpt
  ```

  The following report is generated:

  ```
  set_load 1e-05 [get_nets {sig_out4000}]
  set_load 0.00011 [get_nets {sig_in1000}]
  set_load 0.00011 [get_nets {sig_in1001}]
  set_load 0.00015 [get_nets {node_out6000}]
  ```

6

Flip Chip Commands

This chapter describes the flip chip commands. The commands are presented in alphabetical order.

addAIoFiller

```
addAIoFiller
     {-cell fillerCellName | -cellList {fillerCellName1 fillerCellName2 ...}}
     [-prefix prefix]
     [-aioRowCluster aioRowClusterName | -allAIORowCluster]
     [-setPreplaced]
     [-onlyESD]
     [-onlyShoulder]
     [-onlyGap]
```

Adds area I/O filler cell instances.

The command,

- By default, adds area I/O filler cells to all the area I/O clusters if you do not specify the `-aioRowCluster` and `-allAIORowCluster` parameters.
- Adds filler cells in all the blank sites of the specified area I/O row clusters if you do not specify the `-onlyESD`, `-onlyGap` and `-onlyShoulder` parameters.
- Uses any LEF `CLASS PAD` or `CLASS PAD AREAIO` for adding the area I/O filler cells. By default the command uses LEF `CLASS PAD SPACER` to add filler cells, and also adds filler cells using any LEF `CLASS PAD` displaying a warning in Encounter.
- Does not allow overlapping of filler cells and places non-overlapping filler cells.

Use this command during or after floorplanning. You can also use the command after the placement of manually created area I/Os.

Parameters

`-aioRowCluster` *`aioRowClusterName`*

Specifies the area I/O row cluster name. AIO filler cells are added only to this cluster.

`-allAIORowCluster`

Adds AIOFiller cells to all the area I/O clusters.

`-cell` *`fillerCellName`*

Specifies the name of the area I/O filler cell to be added.

`-cellList {`*`fillerCellName1 fillerCellName2 ...`*`}`

	Specifies the list of area I/O filler cells that fill the gaps in I/O row clusters. Adds the largest filler cell first, followed by smaller filler cells.
`-onlyESD`	Specifies to add ESD (electrostatic discharge) cells with the filler cell name in the area I/O row clusters.
`-onlyGap`	Specifies to add the filler cells with the filler cell name to fill all the gaps in the area I/O row clusters.
`-onlyShoulder`	Specifies to add shoulder cells with the filler cell name in the area I/O row clusters.
`-prefix` *`prefix`*	Specifies the prefix name to be appended for the added area I/O filler cell instance.
`-setPreplaced`	Sets the status of the area I/O filler cells as preplaced.

Example

- The following command fills all area I/O row clusters on ESD, `shoulder`, and `gap` sites with a filler cell `A0_123` with preplaced status:

  ```
  addAIoFiller -cell AO_123 -setPreplaced
  ```

- The following command adds area I/O filler cells, `IOFILL` and `IOFILL_HALF`, to fill all the gaps in all area I/O row clusters:

  ```
  addAIoFiller -allAIORowCluster -onlyGap -cellList { IOFILL IOFILL_HALF }
  ```

addAIORow

```
addAIORow
     -site site_name
     {-bump bumpId | -loc llx lly}
     [-dx x_dist] [-dy y_dist]
     [-orient orient]
     [-H | -V]
     [-esd esites_no...]
     [-shoulder site_no...]
     [-cluster_halo space]
     [-num num]
     [-abutRow]
     [-rowName name]
     [-noSnap]
     [-soft]
     [{-ptnArrX elem1 | -ptnArrY elem2} {-pitchX dist | -pitchY dist}]
```

Adds area I/O rows. Use this command during or after floorplanning.

Parameters

`-abutRow`	When creating two I/O rows that abut each other, this option specifies that one of them will be in a flip orientation to the other.
`-bump` *`bumpId`*	Specifies the bump ID selected in the design window.
`-cluster_halo` *`space`*	Specifies the distance, in microns, around the boundary of a row cluster to prevent other cells from being placed near it.
`-dx` *`x_dist`*	Offsets the reference point, in microns, to the first I/O row on the x axis.
`-dy` *`y_dist`*	Offsets the reference point, in microns, to the first I/O row on the y axis.
`-esd` *`esites_no...`*	Reserves the sites in the row as electrostatic discharge cells (ESD). For example: `-esd 2,7` **Note:** Use only when you create cluster rows with multiple sites. **Note:** Do not use the `-soft` option with this option.
`-H | -V`	Specifies the either a **H**orizontal or a **V**ertical row.

`-loc` *`llx lly`*
: Specifies the lower left x and y coordinate reference points, in microns, of the new I/O row.

`-noSnap`
: Adds the area IO row at the exact location specified by the `addAreaIO -loc` coordinates rather than snapping the row to the standard cell placement grid.

`-orient` *`orient`*
: Specifies the orientation for the row. See Orientation Key on page 303 for more information.
Default: `R0`

`-num` *`num`*
: Specifies the number of sites.

`-ptnArrX` *`elem1`* `|` `-ptnArrY` *`elem2`*
: Specifies the x and y dimensions of the row cluster array size. For example, create a 5 x 8 row array:

```
-ptnArrX 5  -ptnArrY 8
```

`-pitchX` *`dist`* `|` `-pitchY` *`dist`*
: Specifies the distance between bump centers, in microns, in the x and y directions.

`-rowName` *`name`*
: Specifies a user-defined instance name for an added IO Row or a prefix for multiple rows, as in a matrix. If only one row is added, the name is used exactly as specified. If multiple rows are added, a number prefix is appended to the name.

`-site` *`siteName`*
: Specifies the site name for the area I/O row.

`-shoulder` *`site_no`*`...`
: Reserves the sites in a row as shoulder cells. For example:

```
-shoulder 1,10
```

Note: Use only when you create cluster rows with multiple sites.

Note: Do not use `-soft` option with this option.

`-soft`
: Creates a reef or cluster that can be collapsed if no I/O cells are placed in the available spaces within the reef or cluster. Use this parameter if the cluster site is not fully placed so that standard cells can be placed in these spaces if I/O cells are not placed there.

Note: Do not use the `-esd` or `-shoulder` options with this option.

Orientation Key

The following table is a key to row orientation:

Value	Definition	
`R0`		No rotation
`MX`		Mirror through X axis
`MY`		Mirror through Y axis
`R180`		Rotate counter-clockwise 180 degrees
`MX90`		Mirror through X axis and rotate counter-clockwise 90 degrees
`R90`		Rotate counter-clockwise 90 degrees
`R270`		Rotate counter-clockwise 270 degrees
`MY90`		Mirror through Y axis and rotate counter-clockwise 90 degrees

Examples

- The following command creates an I/O row with site name `IO1` at the location related to the center of the bump selected in the design window. There is a 5 x 7 row array and each element in the array is separated by `100` µm in both the x and y axes. Each row has 8 sites.

```
addAIORow -site IO1 -bump 0x33457a -H -num 8 -ptnArrX 5 -ptnArrY 7
   -pitchX 100 -pitchY 100
```

- The following command creates a soft I/O row cluster with a `15.0` micron halo around it:

```
addAIORow -site IO3 -loc 0.0 0.0 -dx 230 -dy 58.8 -num 7 -orient R0 -V
   -soft -abutRow -cluster_halo 15.0
```

- The following command creates an IO Row with a user-defined name:

```
addAIORow -site IO1 -rowName newrow -loc 0 0 -dx 0 -orient R0 -V-num 1
```

addBumpToArrayGrid

```
addBumpToArrayGrid
     -arrayName name
     -bumpName name {-loc X Y | -box X1 Y1 X2 Y2}
```

Adds a bump to the array grid and snaps the bump to the closest grid. Use this command after creating a bump array with the `-arrayName` parameter of the `ciopCreateBump` command.

Parameters

`-arrayName` *name*	Specifies the name of the bump array grid.
`-box` *X1 Y1 X2 Y2*	Creates bumps in the specified area only (all units in microns): ■ *X1*—Lower left x coordinate ■ *Y1*—Lower left y coordinate ■ *X2*—Upper right x coordinate ■ *Y2*—Upper right y coordinate
`-bumpName` *name*	Specifies the name of the bump.
`-loc` *X Y*	Specifies the lower left x and y coordinates of the first bump. Units in microns.

Example

■ The following command adds `bump_1` to `array_1` at location 100.00 100.00:

```
addBumpToArrayGrid -arrayName array_1 -bumpName bump_1 -loc 100.00 100.00
```

assignBump

```
assignBump
     -pio [-maxDistance distance]
     [-pgnet name1 name2 ... | -exclude_pgnet name1 name2 ...
     [-pginst name1 name2 ...]]
     [-area llx lly urx ury]
```

Assigns the bumps closest to the I/O cells.

These bumps can be signal bumps specified in the Verilog netlist or power/ground bumps specified in the command line.

This command is for area I/O flip chip designs. Bumps must be created or loaded from the I/O assignment file.

Use this command after the design is placed.

Note: This command does not affect any existing assignment. Hence, it will not reassign any existing power/ground assignments.

Parameters

`-area llx lly urx ury`	Assigns bumps in the area specified by the lower-left x (*llx*), lower-left y (*lly*), upper-right x (*urx*), and upper-right y (*ury*) coordinates.
`-exclude_pgnet name1 name2 ...`	Specifies the names of the power/ground nets that must be excluded from bump assignment.
`-maxDistance distance`	Specifies the maximum distance, in microns, to look for a bump connection.
`-pginst name1 name2 ...`	Specifies the names of power/ground instances to which the power bumps are assigned.
`-pgnet name1 name2 ...`	Specifies the names of the power/ground nets to which the bumps are assigned. By default, bumps are assigned to all pads whose ports are connected to power/ground nets.

`-pio`	Allows the global router to assign the bumps based on the single layer routing, without any cross overs. However, the bumps must already be assigned using an I/O file or the `assignBump` command. **Note:** Th command reassigns the bumps unless the bump assignment is fixed.

Example

The following command assigns bumps to the power and ground nets, `VDD` and `VSS`, within the area specified by `600.0` (`llx`), `960.0` (`lly`), `2600.0` (`urx`), and `4100.0` (`ury`) coordinates.

```
assignBump -pgnet VDD VSS -area 600.0 960.0 2600.0 4100.0
```

assignIOPinToBump

```
assignIOPinToBump
     -buffer inst_name
     [-pins pinA pinB ...]
     {-tile inst_name | -bump bump_name | -create | -select}
```

Assigns I/O buffers to pins, bumps, or tiles. This command is logged with the object name as the argument and can be cut and pasted for replay. Use this command after bumps are created. It overwrites existing assignments for selected bumps.

Note: The command should be used instead of the assignSigToBump command.

Parameters

`-buffer` *`inst_name`*	Specifies the I/O buffer instance name and its pin(s) that will be assigned to the target tile or bump.
`-bump` *`bump_name`*	Specifies the target bump of the assignment. If multiple pins are specified, the first one will be assigned since only one bump is allowed with this command. To do multiple assignments, use the `-create` or `-select` options of this command.
`-create`	Assigns the specified I/O pins to the tile or bump in the order in which the tile or bump was created.
`-pins` *`pinA pinB ...`*	Specifies the pin(s) of the designated buffer that will be assigned to the tile or bump. This can include multiple pins of the same driver. If omitted, all pins on the driver which are connected to I/O signal (chip's I/O port) will be assigned.
`-select`	Assigns the specified I/O pins to the tile or bump in the selection set.
`-tile` *`inst_name`*	Specifies the target tile instance of the assignment.

Example

- The following command assigns `TEST_IO` buffer with `PAD` pin to tile `bpubv33v12_ca_i4_i24`:

```
assignIOPinToBump -buffer TEST_IO -pins PAD -tile bpubv33v12_ca_i4_i24
```

assignPGBumps

```
assignPGBumps {-selected | -floating} [-V | -H] -nets nameList
```

Assigns power and ground bumps to connect flip chip I/O pins. Use this command after bumps are created. It overwrites existing assignments for selected bumps.

Parameters

`-floating`	Specifies assignment to a floating net.
`-nets` *`nameList`*	Specifies the net names.
`-selected`	Specifies assignment to a selected net.
`-V | -H`	Specifies the style of bump assignment as horizontal (`-H`) or vertical (`-V`), when you define multiple nets using the `-nets` parameter.

Example

- The following command assigns selected bump to power net `VDD`:

  ```
  assignPGBumps -net VDD -selected
  ```

- The following command assigns power/ground bumps to the floating nets, `VSS1, VDD1,` and `VDE1` in the horizontal (`-H`) style:

  ```
  assignPGBumps -H -floating -nets {VSS1 VDD1 VDE1}
  ```

Related Topic

- *Assign Power/Ground Bumps* form in the "Flip Chip Menu" chapter in the *Encounter Menu Reference*.

assignSelectedBump

```
assignSelectedBump
```

Assigns selected bumps to the closest area I/O instances or selected area I/O instances to the closest bumps.

Note: If you select both area I/O instances and bumps, the selected area I/O instances and bumps are assigned to each other.

Parameters

None

Example

- The following command automatically assigns selected bumps to the closest area I/O instances or selected area I/O instances to the closest bumps:

  ```
  assignSelectedBump
  ```

  Note: Before using this command, you must first select the bumps or area I/O instances for assignment.

assignSigToBump

`assignSigToBump` *`ftermPtr`* `{-create | -select | -number | -closest}`

Assigns signal bumps to connect flip chip I/O pins.

Important

It is recommended that you use the assignIOPinToBump command instead of this command.

Parameters

`-create`	Assigns the specified *`fterm`* (signal) to a tile pin or bump by created order.
`-closest`	Assigns to the closest bump. This parameter can only be used after I/O drivers are placed.
`ftermPtr`	Specifies the terminal pointer name.
`-number`	Assigns to the next unassigned bump or tile pin by number.
`-select`	Assigns to the selected bump or tile pin.

Examples

- The following command assigns `CLK` to the closest bump:

  ```
  assignSigToBump CLK -closest
  ```

- The following command assigns I/O signal `HDIN_RN[8]` to bump in created order:

  ```
  assignSigToBump {HDIN_RN[8]} -create
  ```

changeBumpMaster

```
changeBumpMaster
     {-netName netName | -allBumps}
     -bumpMasterName bumpMasterName
     [-fromBumpMasterName bumpMasterName]
```

Enables you to replace the cell master for specified bumps.

Note: Using the edit commands causes bumps to revert back to the original bump. To avoid this, use the removeBumpFromArray command.

Parameters

`-allBumps`	Specifies that all bumps will use the new cell master identified by the `-bumpMasterName` parameter.
`-bumpMasterName` *bumpMasterName*	Specifies the name of the cell master.
`-fromBumpMasterName` *bumpMasterName*	Specifies the name of the cell master from which you want to replace the bump.
`-netName` *netName*	Specifies that all bumps connected to the specified net will use the new cell master. For example, if the *netName* is VDD, all connected bumps will use the cell master identified by the `-bumpMasterName` parameter.

checkBondPadSpacing

`checkBondPadSpacing`

Checks for bond pad spacing violations after using the spaceBondPad command. You can use this command at any time after using the `spaceBondPad` command.

Parameters

None

checkBump

`checkBump -outfile` *`fileName`*

Checks the legality of the bump assignment, and creates a report which is either displayed in the console window or written to a file. Use this command after bumps are assigned with the `assignBump` command or defined in the I/O file with the signal name, but without the `-fixed` constraint.

Parameters

`-outfile` *`fileName`*	Specifies the report filename. If no filename is specified, the report is printed to the console window.

Example

- The following command checks the bump assignment and sends the report to `bump.rpt`:

  ```
  checkBump -outfile bump.rpt
  ```

  A sample report file is as follows:

  ```
  ##################################################################
  # Generated by: Cadence First Encounter 09.10-d028_1
  # OS: Linux i686(Host ID lnx-userID)
  # Generated on: Tue Feb 10 15:36:03 2009
  # Command: checkBump -outfile bump.rpt
  ##################################################################
  Summary of bump assignment in design DTMF_CHIP
  Bump array summary:
  Bump array: array_1 cstagger
  Total bump array is 1
  Bump_100_9_9 array_1 Unassigned
  Bump_99_8_9 array_1 Unassigned
  Bump_98_7_9 array_1 Unassigned
  Bump_97_6_9 array_1 Unassigned
  Bump_96_5_9 array_1 Unassigned
  Bump_95_4_9 array_1 Unassigned
  Bump_94_3_9 array_1 Assigned port_pad_data_out[8] IOPADS_INST/Ptdspop08
  Bump_93_2_9 array_1 Assigned tdigit[0] IOPADS_INST/Ptdigop0
  Bump_92_1_9 array_1 Assigned port_pad_data_in[8] IOPADS_INST/Ptdspip08
  Bump_91_0_9 array_1 Assigned tdigit[4] IOPADS_INST/Ptdigop4
  ```

```
Bump_90_9_8 array_1 Unassigned
```

ciopCreateBump

```
ciopCreateBump bumpcell
     [-name bump_name]
     [-in_cell cellptr]
     [-loc X Y | -in_box X1 Y1 X2 Y2]
     [ [-edge_spacing X [Y]] -pitch X [Y]
     [-array [MxN] [-stagger] | -perim N [-center MxN [-cstagger]]] ]
     [-arrayName name]
```

Creates bumps and instances of bump cells.

Parameters

`-array` *M* x *N*	Creates an *M* column x *N* row bump array on the whole chip area. Specify the number of columns (M) and the number of rows (N).
`-arrayName` *name*	Specifies the name of the bump array being created. By assigning a name to the bump array, Encounter is able to save the parameters that were used to generate the bump array. This provides you with the means to access the bump array and modify it as needed for use in a future or current session of Encounter. The following text commands enable you to modify a bump array: ■ addBumpToArrayGrid ■ deleteBumpArray ■ editBumpArray ■ selectBumpArray
bumpcell	Specifies the name of the bump cell.
`-center` *M* x *N*	Centers an *M* column x *N* row bump array.
`-cstagger`	Centers bumps in stagger (offset) style.
`-edge_spacing` *X* [*Y*]	

	Specifies a minimum x and y distance, in microns, to the edge of the chip from the center of the outermost bumps. The edge spacing constraint is applied starting from the lower-left corner and stepping toward the upper right. The spacing of the edge of the right-most bump will be at least the specified value or more.
`-in_cell` *`cellptr`*	Creates bumps inside the specified cell. *Default:* Current top cell
`-loc` *`X Y`*	Specifies the x and y coordinates of the center of the first bump. Units are in microns.
`-in_box` *`X1 Y1 X2 Y2`*	Creates bumps in the specified area only (all units microns): ■ *`X1`*—Lower left x coordinate ■ *`Y1`*—Lower left y coordinate ■ *`X2`*—Upper right x coordinate ■ *`Y2`*—Upper right y coordinate
`-name` *`bump_name`*	Specifies the bump name. For more than one bump, the name will be generated automatically.
`-perim` *`N`*	Creates *`N`* rows of bumps inside the chip boundary.
`-pitch` *`X`* `[`*`Y`*`]`	Specifies the distance between bump centers, in microns, in the x and y directions.

`-stagger`	Staggers (offsets) the bump array. For example:
	`******`
	` ******`
	`******`

Example

■ The following command creates a `100` x `100` bump array with a pitch of `200` microns:

```
ciopCreateBump BUMPCELL -loc 100.00 100.0 -pitch 200 200 -array 100x100
```

ciopCreateBumpCell

```
ciopCreateBumpCell bumpCell_name
     {-box x1 y1 x2 y2 ... | -polygon x1 y1 x2 y2...}
     [-layer num]
```

Creates a master bump cell definition.

Parameters

`-box x1 y1 x2 y2...`	Specifies the box coordinates, in microns, of the bump cell: ■ *X1*—Lower left x coordinate ■ *Y1*—Lower left y coordinate ■ *X2*—Upper right x coordinate ■ *Y2*—Upper right y coordinate **Note:** More than one bump box can be created. Each box must have two x coordinates and two y coordinates.
`-layer num`	Specifies the layer of the bump cell.
`-polygon x1 y1 x2 y2...`	Specifies the polygon vertices, in microns, of the bump cell.

Example

- The following command creates a `40` micron square bump cell in layer `6`:

  ```
  ciopCreateBumpCell BUMPCELL -box 0 0 40 40 -layer 6
  ```

ciopLoadBumpColorMapFile

```
ciopLoadBumpColorMapFile
     fileName
```

Loads the bump color map file to specify colors for bumps. The command supports all the 256 colors specified in the `rgb.txt` file in `/usr/openwin/lib/X11`.

Parameters

fileName	Specifies the name of the bump color map file to load.

Example

- The following command loads the bump color map file `BumpColor.Map` and assigns colors to bumps in the design:

  ```
  ciopLoadBumpColorMapFile BumpColor.Map
  ```

  Output:

  ```
  port_pad_data_out[8] wheat
  tdigit[1] green
  tdigit[2] green
  ```

deleteAIoFiller

```
deleteAIoFiller
     -cell fillerCellName
     [-prefix prefix]
     -aioRowCluster aioRowClusterName | -allAIORowCluster
     [-Inst fillerInstanceName]
```

Deletes area I/O filler cell instances. Use this command after using the addAIoFiller command.

Parameters

`-aioRowCluster` *aioRowClusterName*

Specifies the area I/O row cluster name where all area I/O filler cells will be deleted.

Note: This parameter overrides `-allAIORowCluster`, i.e If you specify both, `-aioRowCluster` *aioRowClusterName* and `-allAIORowCluster` parameters, then only `-aioRowCluster` *aioRowClusterName* is valid.

`-allAIORowCluster` Deletes all area I/O filler cells from all area I/O row clusters.

`-cell` *fillerCellName*

Specifies the name of the area I/O filler cell to delete.

`-Inst` *fillerInstanceName*

Specifies the name of the area I/O filler instance name to delete.

`-prefix` *prefix* Specifies the prefix name of the area I/O filler cell instance to be removed.

Example

- The following command deletes a filler cell from all area I/O row clusters:

  ```
  deleteAIoFiller -cell AO_123 -allAIORowCluster
  ```

deleteBumpArray

`deleteBumpArray` *`name`*

Deletes a bump array.

Note: You cannot delete fixed bumps with this command. Use this command after creating a bump array with the `-arrayName` parameter of the `ciopCreateBump` command.

Parameters

`name`	Specifies the name of the bump array.

Example

- The following command deletes bump array `array_1` from the current design:

```
deleteBumpArray array_1
```

deleteBumps

```
deleteBumps
     {-all |
     -selected |
     -overlap_blockages |
     -overlap_macro |
     -overlap_areaio |
     -unassigned |
     -floating}
```

Removes bumps from the design.

Parameters

`-all`	Deletes all bumps.
`-floating`	Deletes bumps which are not assigned to any nets.
`-overlap_areaio`	Deletes bumps that overlap with the area I/O cells.
`-overlap_blockages`	Deletes all bumps that overlap routing blockages on the same layer. For example, if a bump on layer M7 overlaps a routing blockage on layer M7, the bump is deleted when you use this parameter.
`-overlap_macro`	Deletes all bumps that overlap the selected macros. You must first select the macros in the design display area, then use this command to delete the bumps.
`-selected`	Deletes selected bumps. You must first select the bumps in the design display area, then use this command to delete the bumps.
`-unassigned`	Deletes all the unassigned bumps in the design. Specify this option after assigning and optimizing all the bumps in the design to clean up the database.

Example

- The following command deletes all of the bumps in the current design:

  ```
  deleteBumps -all
  ```

editBumpArray

```
editBumpArray -arrayName name
     [-pitch X Y]
     [-edge_spacing X Y]
     [-loc X Y | -in_box X1 Y1 X2 Y2]
     [-array [MxN] [-stagger | -nostagger] | -perim N [-center MxN [-cstagger | -
     nocstagger]]]
```

Edits a bump array. Use this command after creating a bump array with the `-arrayName` parameter of the `ciopCreateBump` command.

Parameters

`-array [MxN]`	Creates an M column x N row bump array on the whole chip area, based on the number of columns (M) and the number of rows (N) you specified.
`-arrayName` *name*	Specifies the name of the bump array. ■ If all of the bumps in the specified bump array are unassigned, the `editBumpArray` command deletes them and creates new bumps for the bump array based on the new parameters. ■ If any of the bumps in the specified bump array are assigned, the `editBumpArray` command unassigns them, deletes them, then creates a new bump array. ■ If any of the bumps in the specified bump array are routed, the `editBumpArray` command generates an error. You must delete the routing before editing the bump array.
`-center MxN`	Centers an M column x N row bump array.
`-cstagger`	Centers the bumps in the bump array in stagger (offset) style.

`-edge_spacing` *`X Y`*

Specifies a minimum x and y distance to the edge of the chip from the center of the outermost bumps. Units in microns. The edge spacing constraint is applied starting from the lower-left corner and stepping toward the upper right. The spacing of the edge of the right-most bump will be at least the specified value or more.

`-in_box` *`X1 Y1 X2 Y2`*

Creates bumps in the specified area only (units in microns):

- *`X1`*=Lower-left *x* coordinate
- *`Y1`*=Lower-left *y* coordinate
- *`X2`*=Upper-right *x* coordinate
- *`Y2`*=Upper-right *y* coordinate

`-loc` *`X Y`*

Specifies the lower-left *x* and *y* coordinates of the first bump. Units in microns.

`-nocstagger`

Does not center the bumps in the bump array in stagger style.

`-nostagger`

Unstaggers the bump array.

`-perim` *`N`*

Creates *`N`* rows of bumps inside the chip boundary.

`-pitch` *`X Y`*

Specifies the distance between bump centers in the *`X`* and *`Y`* directions. Units in microns.

`-stagger`

Staggers (offsets) the bump array.

For example:

```
******
 ******
******
```

fixBondPad

```
fixBondPad
     {-ioInstName InstName [-pinName pinName] | -selected}
```

Assigns a fixed status to a specified bond pad. This restricts other bond pad commands from modifying the bond pad.

The following commands provide additional support for the staggering of bond pads:

- getBondPad
- placeBondPad
- staggerBondPad
- unfixBondPad

Parameters

`-ioInstName` *`instName`*	Specifies the name of the I/O instance whose bond pad will be assigned as fixed.
`-pinName` *`pinName`*	Specifies the name of the pin inside the instance.
`-selected`	Assigns a fixed status to a selected bond pad. You must first select the bond pad in the design display area, then use this parameter to assign the fixed status.

Example

- The following command assigns a fixed status to the bond pad in I/O instance `pad_addr[30]`:

  ```
  fixBondPad -ioInstName {pad_addr[30]}
  ```

getBondPad

```
getBondPad
     -ioInstName InstName
     [-pinName pinName]
```

Gets the current stagger position of a bond pad on a specified I/O instance.

The following commands provide additional support for the staggering of bond pads:

- fixBondPad
- placeBondPad
- staggerBondPad
- unfixBondPad

Parameters

`-ioInstName` *`instName`*	Specifies the name of the I/O instance whose bond pad position is reported.
`-pinName` *`pinName`*	Specifies the name of the pin inside the instance.

Example

- The following command gets the current stagger position of the bond pad on I/O instance `pad_addr[30]`:

```
getBondPad -ioInstName {pad_addr[30]}
Bond Pad Name = BUMPIN_337, Position = I
```

getFlipChipMode

```
getFlipChipMode
     [-connectPowerCellToBump]
     [-constraintFile]
     [-extraConfig]
     [-layerChangeBotLayer]
     [-layerChangeTopLayer]
     [-multipleConnection]
     [-routeWidth]
```

Returns the following information about a specified `setFlipChipMode` parameter in the Encounter log file and in the Encounter console.

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software returns information for all of the `setFlipChipMode` mode parameters.

Parameters

`parameter_names`	Returns information for the specified parameters. You can specify one or more parameters. See `setFlipChipMode` for descriptions of the flip chip mode parameters you can specify.
`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setFlipChipMode` parameters in Tcl list format.

handlePtnAreaIo

```
handlePtnAreaIo
     [-insertBuffer bufferName | -noInsertBuffer]
     [-top extension]
     [-bottom extension]
     [-left extension]
     [-right extension]
     [-selectedCell]
     [-pinOnBoundary]
```

Specifies the name of the buffer to be used for feedthrough insertion in a flip chip design. Use this command before partitioning the flip chip design.

Parameters

`-bottom` *`extension`*
: Specifies how far, in db units, to expand the Area I/O cell box in the bottom direction when determining the size of the placement and routing obstructions to be pushed down.

`-insertBuffer` *`bufferName`* | `-noInsertBuffer`
: Specifies either the name of the buffer, or that there are no buffers to be inserted.
Default: `-noInsertBuffer`

`-left` *`extension`*
: Specifies how far, in db units, to expand the Area I/O cell box in the left direction when determining the size of the placement and routing obstructions to be pushed down.

`-pinOnBoundary`
: Specifies the internal pin created to be placed at the intersection of the hole punch (placement obstruction) and the space reserved for the partition. If it is not specified, the default is that the internal pin is placed directly on top of the area I/O cell.

`-right` *`extension`*
: Specifies how far, in db units, to expand the Area I/O cell box in the right direction when determining the size of the placement and routing obstructions to be pushed down.

`-selectedCell`
: Specifies hole punches for all currently selected cells that overlap partitions if they are Area I/O cells. If you do not specify this parameter, the command specifies hole punches for all Area I/O cells that overlap partitions.

`-top` *`extension`*	Specifies how far, in db units, to expand the Area I/O cell box in the top direction when determining the size of the placement and routing obstructions to be pushed down.

ioInstOverlapCheck

`ioInstOverlapCheck`

Checks to see if any area I/O instance is overlapped with another area I/O instance. If an overlap occurs, a warning message is dumped to the console window, and the overlapped instances are highlighted in the work area. Use this command before flattening the tiles.

Parameters

None

placeAIO

```
placeAIO
     [-onlyAIO]
     [-assignBump]
     [-maxDistance distance]
     [-fast]
     [-packing]
     [-ignoreAIOByName {list}]
     [-ignoreAIOByCellName {list}]
```

Places I/O driver cells.

The area I/O Library must be loaded when importing the design. Area I/O rows are also required.

Note: Once you run the `placeAIO -onlyAIO` command to place the area I/Os, you can run `placePIO -cellList` to specify the cell(s) that must be placed on the periphery.

Parameters

`-assignBump`	If you have unassigned bumps for area I/O instance connections, this connects area I/O instances to the nearest unassigned bumps. This option requires that bump cells and bump locations are already loaded. **Note:** You can also assign bumps after area I/O placement by using the `assignBump` command.
`-fast`	Runs area I/O placement in the fast mode, trading speed for quality.
`-ignoreAIOByCellName {list}`	Ignores all area I/O instances that instantiate from the cells specified in the list, during placement. For example, if a cell `PDI` has 2 AIO instances, `test_in_1` and `test_in_2`, this parameter ignores both the instances during placement.
`-ignoreAIOByName {list}`	Ignores all the area I/O cells specified in the list, during placement. Specify this parameter if you already placed the area I/O cells on the periphery and you want to avoid placing the same cells in the core.
`-maxDistance value`	Specifies the maximum allowable distance between the I/O driver cells and the bumps.

`-onlyAIO`	Places only area I/O instances. If this option is not specified, the command will also place all standard cell instances and blocks.
`-packing`	Specifies the packing of I/O driver cells within each cluster of rows. This option is only applicable to designs with clusters of I/O rows.

Examples

- The following command places area I/O instances and all standard cell instances, and assign bumps for all chip I/O pins:

  ```
  placeAIO -assignBump
  ```

- The following command places only area I/O cells:

  ```
  placeAIO -onlyAIO
  ```

placeBondPad

```
placeBondPad
     {-ioInstName instName | -selected}
     [-pad padName]
     [-pinName pinName]
     [-position {i | m | o}]
     [-fix]
```

Places a bond pad on a specified I/O instance. It defines the stagger position of a specified bump on a single or selected set of I/O instances.

The following commands provide additional support for the staggering of bond pads:

- fixBondPad
- getBondPad
- staggerBondPad
- unfixBondPad

Parameters

`-fix`	Assigns a fixed status to the specified bond pad. This restricts other bond pad commands from modifying the bond pad.
`-ioInstName` *name*	Specifies the name of the I/O instance of where the bond pad will be placed.
`-pad` *padName*	Specifies the name of the bond pad.
`-pinName` *pinName*	Specifies the name of the pin.
`-position {I | M | O}`	Defines the stagger position of the specified bond pad on the specified I/O instance. You can stagger the position of the bond pad using the inner (`I`), middle (`M`), and outer (`O`) positions.
`-selected`	Places a selected bond pad on the specified I/O instance. You must first select the bond pad in the design display area, then use this parameter to place the bond pad on the specified I/O instance.

placePIO

```
placePIO
     [-assignBump]
     [-optIOs]
     [-overflowMap]
     [-maxIOHeight value]
     [-ioFile fileName]
     [-rdlConstraintFile fileName]
     [-noRandomPlacement]
     [-extraConfig fileName]
     [-cellList {cellList}]
     [-instList {instList}]
     [-fixPadSide]
     [-fixNetPadSide]
```

Places CLASS PAD and CLASS PAD AREAIO cells on the die boundary (periphery) in random order.

The command reads options specified in the `setFlipChipMode` command before placing the CLASS PAD and the CLASS PAD AREA IO cells on the die boundary. It retrieves the `rdlconstraintfile` and `sroute` config file information from the `setFlipChipMode -constraintFile` and `setFlipChipMode -extraConfig` command options.

During bump assignment, the command uses resistance constraints that are specified in the `rdlconstraintfile` for net(s). To view a constraint file, see Examples and Report Files in "Flip Chip Methodologies" chapter of the *Encounter User Guide*.

The `placePIO -optIos -assignBump` command also reads `setFlipChipMode -routeWidth` value before reassigning the signal bumps in the design.

Use this command after the design is loaded.

Note: IO cells must be defined as CLASS PAD on the boundary of the design. The peripheral I/O Library must be loaded when importing the design. Peripheral I/O rows are also required.

Parameters

`-assignBump`	Allows signal bumps to be reassigned to improve routing if the I/O cells have been fixed.

`-cellList {instlist}`

Specifies which area I/O cell(s) must be placed on the periphery.

Once you run `placePIO -cellList` to grab the area I/O cells for placement on the periphery, you can run `placeAIO -ignoreAIOByName {`*`list`*`}` to avoid the placement of these cells in the core area.

`-extraConfig` *`fileName`*

Specifies a file containing user-defined placement options written in `sroute` syntax. For more information, see Extra Configuration File Options for fcroute.

`-fixNetPadSide` Assigns a fixed status to a specified net pad side.

`-fixPadSide` Assigns a fixed status to a specified I/O pad side.

`-ioFile` *`fileName`* Specifies the name of a file to control the placement of the I/O cells.

`-instList {`*`instList`*`}`

Specifies which area I/O instance(s) must be grabbed for placement on the periphery.

Once you run `placePIO -instList` to place the area I/O instances on the periphery, you can run `placeAIO -ignoreAIOByName {`*`list`*`}` to avoid the placement of these instances in the core area.

`-maxIOHeight` *`value`* Specifies the maximum height for the I/O row.

`-noRandomPlacement` Without options, the `placePIO` command randomly places all I/O cells on the periphery. This option inhibits this behavior and optimizes pads without initial placement.

`-optIOs` Allows I/O cells to be moved to improve routing if the bumps have been fixed optimizing the initial placement.

`-overflowMap`

Displays the congestion map using the width and spacing parameters specified in the extra configuration file.

To display the congestion map, run the following command to specify the options for the extra configuration file:

```
placePIO -noRandomPlacement -overflowMap -extraConfig
scripts/global.cfg
```

where:

`scripts/global.cfg`:

```
srouteBottomLayerLimit 8 /* Assume the intended routing
layer range is M8 to M9 */
srouteTopLayerLimit 9
srouteRouteWidth 10000 /* n is the routing width, in db
unit - 5u */
```

`-rdlConstraintFile` *`fileName`*

Specifies a file with constraints to control the placement of the I/O cells.

The command uses resistance constraints specified in this file, for cell placement.

To view a constraint file, see "Examples and Report Files" in Flip Chip Methodologies chapter of the *Encounter User Guide*.

Related Topics

- *Flip Chip Route – Advanced – Routing Constraints GUI* in "Route Menu" chapter of the *Encounter Menu Reference*.
- Routing and Placement Constraints in "Flip Chip Methodologies" chapter of the *Encounter User Guide*.

removeBumpFromArray

```
removeBumpFromArray
     {-selected | -bumpInstName instName}
```

Removes a specified bump from its assigned bump array grid. After the bump is removed from the bump array grid, you can move the bump off of the bump array grid and have it snap to the manufacturing grid.

Note: Once the bump has been removed from the bump array, you can no longer edit the bump using the editBumpArray command.

Parameters

`-bumpInstName` *instName*	Specifies the name of the bump to remove from the bump array.
`-selected`	Removes a bump from its assigned bump array. You must first select the bumps in the design display area, then use this parameter to remove them from the bump array.

Example

- The following command removes BumpA from its assigned bump array grid:

  ```
  removeBumpFromArray -bumpInstName BumpA
  ```

reportProbePins

```
reportProbePins
```

Generates a report of all probe pins that are used for package-level design and testing. If no probe points exist, the software returns the following message:

```
“No Probe Points Defined”
```

Note: You can set or delete probe points using the setProbePin or unsetProbePin commands respectively.

Example

- The following command generates a report of all probe pins:

  ```
  reportProbePins
  ```

reportSpecialRoute

```
reportSpecialRoute
     reportName
     [-wirelength]
     [-highlight45layer layerNumber]
     [-lenghtthreshold value]
     [-net netName]
```

Reports the length of all RDL routes (SPECIALNETS) created with the `fcroute` command. This includes support for Manhattan and 45-degree routes. You can report the wire length of each SPECIALNET and report the SPECIALNETS that are greater in length than the specified threshold value.

Parameters

`-highlight45layer` *layerNumber*

Highlights (in yellow) all nets with 45-degree routing on the specified layer in the design display area.

`-lengththreshold` *value*

Reports on the SPECIALNETS that are greater in length than the specified value. These values are listed in the output file specified by *reportName* and the routes are highlighted in the GUI.

`-net` *netName*

Reports on the wire length of each SPECIALNET for a specified net.

Note: You must use this parameter in conjunction with the `-wirelength` or `-lengththreshold` parameters.

reportName

Specifies the report file name.

`-wirelength`

Reports on the wire length of each SPECIALNET.

Example

- The following command reports the wire length of each SPECIALNET and lists it in the `spwire.rpt` report file:

  ```
  reportSpecialRoute spwire.rpt -wirelength
  ```

  Example report file:

  ```
  ##############################################################
  # Generated by: Cadence First Encounter 08.10-b020_1
  ```

```
# OS: Linux x86_64(Host ID icdopt26s)
# Generated on: Fri Jan 11 11:13:26
# Command: reportSpecialRoute spwire.rpt -wirelength
#####################################################################
NET: imid TOTAL SEGMENTS 2 WIRE LENGTH 132.1600 VIAS 0
WIDTH 8.0000 LENGTH 132.1600
IOWIRE SEGMENTS 2 WIRE LENGTH 132.1600 VIAS 0
```

selectBumpArray

`selectBumpArray` *`name`*

Selects all bumps in a specified bump array. You can use the Attribute Editor to query a single bump by double-clicking on the left mouse button. Use this command after creating a bump array with the `-arrayName` parameter of the `ciopCreateBump` command.

Parameters

`name`	Specifies the name of the bump array.

Example

- The following command selects bump array `array_1` in the current design:

```
selectBumpArray array_1
```

setBumpFixed

```
setBumpFixed {-allBumps | -byBumpName {list} | -byBumpSite {list}}
```

Fixes a specified group of assigned bumps to keep them from being reassigned. The software updates the DEF file by adding the +FIXEDBUMP attribute to the specified group of assigned bumps to indicate that they are now in a fixed state. You can use the defOut command to view the bumps with the +FIXEDBUMP attribute. Use this command after bumps are assigned with the assignBump command or defined in the I/O file with the signal name.

Parameters

`-allBumps`	Fixes all of the assigned bumps.
`-byBumpName` *list*	Fixes the assigned bumps that are specified in the bump name *list*.
`-byBumpSite` *list*	Fixes the assigned bumps that are specified in the bump site *list*.

Example

- The following command fixes all of the assigned bumps:

  ```
  setBumpFixed -allBumps
  ```

setBumpPlacementStatus

```
setBumpPlacementStatus
     [-help]
     [-bumpName Name | -selected]
     PLACED | FIXED | COVER
```

Specifies the bump placement status.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `setBumpPlacementStatus` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man setBumpPlacementStatus`.
`-bumpName` *`Name`*	Specifies the name of the bump to set the placement status.
`PLACED | FIXED | COVER`	Specifies the bump placement status.
`-selected`	Specifies to set the bump placement status for the selected bump.

Example

- The following command sets the placement status of the bump `Bump_53_2_5` to `FIXED`:

  ```
  setBumpPlacementStatus Bump_53_2_5 FIXED
  ```

  Note: You can also double-click the bump to change the placement status from `PLACED` to `FIXED`.

Related Topics

- "Data Preparation" chapter of the *Encounter User Guide*
  - Defining Bump Cell Placement Status

setFlipChipMode

```
setFlipChipMode
     [-help]
     [-reset]
     [-connectPowerCellToBump {true | false}]
     [-constraintFile fileName]
     [-extraConfig fileName]
     [-layerChangeBotLayer layerNum]
     [-layerChangeTopLayer layerNum]
     [-multipleConnection {multiPadsToBump | multiBumpsToPad | default}]
     [-routeWidth real]
```

Loads data for placePIO and fcroute commands. Both, placePIO -optIOs -assignBump -noRandomPlacement and fcroute commands use the same data for I/O optimization, bump assignment, and flip chip routing.

Important

You need to first execute setFlipChipMode command followed by placePIO, and fcroute commands.

Use the getFlipChipMode command to return the current settings for the setFlipChipMode command.

Parameters

-help	Outputs a brief description that includes type and default information for each setFlipChipMode parameter.
-reset	Resets parameters to their default values. The -reset parameter *must* be the first parameter specified. If you specify -reset by itself, the software resets all setFlipChipMode parameters to their default values. If you specify parameters after -reset, the software resets only those parameters to their default values.
-connectPowerCellToBump {true | false}	Connects all power bumps to the I/O cell pin. It does not connect to a power or ground stripe or ring. *Default*: false
-constraintFile *fileName*	

Specifies the name of the file that contains the names of the nets that are to have differential routing. If you do not specify this parameter and if no differential pairs are defined in the PROPERTY statement of the DEF file, differential routing is not done. For more information, see Creating Differential Routing to Signal Bumps in the *Encounter User Guide.*

`-extraConfig` *`fileName`*

Specifies the name of an extra configuration file that uses options defined in `sroute`.

`-layerChangeBotLayer` *`layerNum`*

Specifies the bottom-most metal layer that the software can use when routing bumps.

Default: If you do not specify this parameter, the top 2 layers are used.

`-layerChangeTopLayer` *`layerNum`*

Specifies the top-most metal layer that the software can use when routing bumps.

Default: Top layer.

`-multipleConnection {multiPadsToBump | multiBumpsToPad | default}`

Specifies routing connections between multiple pads and bumps.

- `multiPadsToBump`

  Enables routing from multiple pads to one bump in parallel.

- `multiBumpsToPad`

  Enables routing from multiple bumps to one pad in parallel.

- `default`

  Enables routing from one bump to one pad in parallel.

`-routeWidth` *`real`* Connects wires to bumps using the specified route width not wider than the I/O pad pin.

Example

The following command specifies the flip chip constraints in the constraint file `CFG/res.constr` enabling routing from a single bump to single power cell in parallel, having a route width of `10` micrometers on the metal layer `8`:

```
setFlipChipMode -connectPowerCellToBump true -constraintFile CFG/res.constr -
extraConfig  CFG/ex.cfg -layerChangeBotLayer 8 -layerChangeTopLayer 8 -
multipleConnection default -routeWidth 10
```

setProbePin

```
setProbePin {-instName instName -pinName pinName | -bump bumpName}
```

Sets a pad or bump as a probe point. Probe points are used for package-level design and testing. To delete or report on probe points, use the unsetProbePin or reportProbePins commands respectively.

Parameters

`-bump` *`bumpName`*	Specifies the instance name of the bump.
`-instName` *`InstName`*	Specifies the name of the I/O instance.
`-pinName` *`pinName`*	Specifies the name of the pin on the I/O instance that will be set as a probe point.

Example

- The following command sets pin PAD of I/O instance Bump_1 as a probe point:

  ```
  setProbePin -instName Bump_1 -pinName PAD
  ```

spaceBondPad

```
spaceBondPad
     [-spread]
```

Spaces bond pads and the corresponding area I/O instances based on the constraints returned by the `getCellDist()`Tcl script.

The command distributes excess space until all the bond pads reach their maximum spacing constraint returned by the `getCellDist ()` Tcl script or until the time when no extra space is left between the instances.

The initial distribution of space occurs for the clusters of instances that have the first or the last bond pads in the row. The remaining iterations equally distribute the remaining space between each I/O. The I/O instances that are in the end are placed at minimum spacing to the die edges, both on the left and right sides of the die.

The I/O pads will snap to I/O sites and the corresponding bond pads (bumps) will shift accordingly.

Note: To check the bond pad spacing after using this command, use the checkBondPadSpacing command.

You can define the constraints in the `getCellDist()`Tcl script as follows:

```
proc getCellDist { xp yp xn yn padPosition padCount side ioRow staggerPos
ioName }
```

where:

xp yp	Specifies the x and y coordinates of a previous bond pad having the same stagger position. If first, the location is `0, 0`.
xn yn	Specifies the x and y coordinates of the previous neighboring bond pad having any stagger position. If first, the location is `0, 0`.
padPosition	Specifies the position of the bond pad. The position can be `i` (inner), `m` (middle), or o (outer).
padCount	Specifies the current bond pad count for bond pads at the same stagger position. For example, 1 for the first inner bond pad, 2 for the second inner bond pad, and so on.
side	Specifies the n (North), s (South), e (East), or `w` (West) side of the bond pad.

ioRow	Specifies the I/O row.
staggerPos	Specifies the stagger type.
ioName	Specifies the name of the I/O.

- The bond pad spacing for North and South is done from left to right.
- The bond pad spacing for East and West is done from bottom to top.
- The bond pad spacing for each side and ring is done independently.

Parameters

`-spread`	Removes any extra spacing that may exist between I/O cells, which were placed based on the rules defined in the `getCellDist()`Tcl script, and evenly distirbutes the extra space between all of the I/O cells in that row. You can use this parameter if there are large gaps (spaces) between I/O cells.

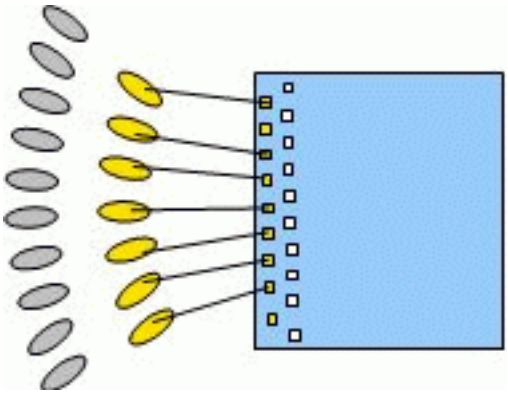
staggerBondPad

```
staggerBondPad
     {-startIoInstName instName [-endIoInstName instName] | -ring number |
     -side {e | w | n | s} [-ring number] | -all}
     {-pattern pattern_string | -startStaggerPosition {i | m | o}}
     [-pad position padName position padName...]
     [-pinName pinName]
```

Generates a staggered wirebond pad pattern on top of an existing I/O ring. In this design methodology, bond pads are separate LEF macro definitions from the I/O driver cells, but are placed on top of the I/O driver cells to form the combined bond pad and I/O driver circuitry.

The command supports both, `CLASS PAD` and `CLASS PAD AREAIO` instances.

To produce the staggered pattern, multiple bond pad cells are defined in the LEF for each I/O cell, each with its bond pad in a specific location corresponding to an outer, middle, or inner stagger position. The bond pads are defined such that when placed on the I/O cell, any one of them will connect to the I/O cell pin with no or minimal RDL routing.



Encounter generates a stagger pattern by selecting the outer, middle, or inner stagger bond pad to place on each I/O cell based on the location of that I/O instance in the I/O ring and the pattern that is being generated. If the order of the I/O cells is changed, or if cells are added or deleted, the stagger pattern may be regenerated. Because the bond pad macros are separately defined, they can be easily swapped with bond pads of different stagger positions without changing the logic of the design.

To define the relationship between the bond pads and I/O cells in the LEF file, the following properties must be defined in each LEF file where they are referenced:

```
PROPERTYDEFINITIONS
    PIN bondPadOuter STRING;
    PIN bondPadMiddle STRING;
    PIN bondPadInner STRING;
```

```
    PIN ioCellOriginX REAL;
    PIN ioCellOriginY REAL;
    MACRO ioCellOffsetX REAL;
    MACRO ioCellOffsetY REAL;
END PROPERTYDEFINITIONS
```

The LEF macros define the property values as follows:

```
MACRO IOCELL
    CLASS PAD AREAIO ;
    ...
    SIZE 50.0 BY 225 ;
    ...
    PIN PAD
        PROPERTY bondPadOuter "BUMPA BUMPA1 BUMPD" ;
        PROPERTY bondPadMiddle "BUMPB" ;
        PROPERTY bondPadInner "BUMPC" ;
        PROPERTY ioCellOriginX 25.0 ;
        PROPERTY ioCellOriginY 0.0 ;
    END PAD
END IOCELL
MACRO BUMPA
    CLASS COVER BUMP
    ...
    PROPERTY ioCellOffsetX -40 ;
    PROPERTY ioCellOffsetY -70 ;
END BUMPA
```

Encounter automatically names the stagger bond pads as: *IoCellInstName*_PAD*#*.

where:

IoCellInstName	Specifies the instance name of the I/O cell on which the bond pad is being placed.
#	Specifies the incremented bond pad number. The number 1 is used for all single pad I/O cells. This number is incremented when there are multiple bond pads on a single I/O cell. For example, if I/O cell CKdiff contained two pins, two bond pads would be placed on this cell and named as CKdiff_PAD1 and CKdiff_PAD2.

The following commands provide additional support for the staggering of bond pads:

- fixBondPad
- getBondPad
- placeBondPad
- unfixBondPad

Parameters

`-all`	Specifies that bond pads will be placed on all I/O instances.
`-endIoInstName` *`instName`*	Specifies the name of the last I/O instance of where bond pads will be placed. You can use this parameter in conjunction with the `-startIoInstName` parameter. **Note:** Bond pads are placed in a clock-wise pattern from the starting I/O instance (`-startIoInstName`) to the last I/O instance (`-endIoInstName`).
`-pad` *`position padName position padName...`*	Specifies which bond pad and position to use when multiple bond pads are specified in the LEF file for a given inner (i), middle (m), or outer (o) position. For example, If there are three outer bond pads specified in the LEF file, you can use this parameter to choose which bond pad to use as the outer bond pad. *Default*: The software uses the first bond pad in the list of bond pads for a given position.
`-pattern` *`pattern_string`*	Specifies the stagger pattern of the bond pads on the I/O instances. *`pattern_string`* specifies any combination of `i` (inner), `m` (middle), and o (outer).
`-pinName` *`pinName`*	Specifies the name of the pin inside the instance. If the instance specified by the `-startIoInstName` parameter contains multiple pins, you use this parameter to specify which pin to start with.

`-ring` *`number`*
: Specifies which ring to operate on with ring 1 being the outer most ring. The ring number is incremented by one as you get closer to the core. For a three-ring design, the inner most ring is ring 3.

`-side {e | w | n | s}`
: Specifies the geographical side of the die. You can specify the East (`e`), West (`w`), North (`n`), or South (`s`) side of the die.

`-startIoInstName` *`instName`*
: Specifies the name of the first I/O instance of where bond pads will be placed. You can use this parameter in conjunction with the `-endIoInstName` parameter. If multiple pins exist in the instance, you can use the `-pinName` parameter to specify which pin to start with.

  Note: Bond pads are placed in a clock-wise pattern from the starting I/O instance (`-startIoInstName`) to the last I/O instance (`-endIoInstName`) if applicable.

`-startStaggerPosition {i | m | o}`
: Specifies the starting stagger position of the specified bond pad (`-pad`) on the specified I/O instance (`-startIoInstName`). You can start the stagger position of the bond pad using the inner (`i`), middle (`m`), or outer (`o`) position. The default stagger position sequence is inner (`i`), middle (`m`), and outer (`o`). If you start with a stagger position of middle (`m`), the next bond pad is placed in an outer (`o`) stagger position followed by a bond pad placed in an inner (`i`) stagger position.

Example

- The following command places bond pads on all I/O instances using a stagger pattern of inner, outer, and middle and using bond pad `BUMPA1` at the outer position:

  ```
  staggerBondPad -all -pattern iom -pad o BUMPA1
  ```

- The following command places bond pads on the I/O instances of the outer most ring (East side) starting with a bond pad placed in the middle stagger position:

  ```
  staggerBondPad -side e -startStaggerPosition m -ring 1
  ```

- The following command places bond pads from I/O instance `abc` to I/O instance `xyz`, starting with pin `PAD` of I/O instance `abc` and using an inner, inner, outer stagger pattern:

```
staggerBondPad -startIoInstName abc -endIoInstName xyz -pinName PAD
-pattern iio
```

swapSignal

`swapSignal`

Swaps signal between the 2 selected instances (I/O pads or bumps).

Parameters

None.

unassignBump

```
unassignBump
     {-allBumps | -byBumpName {list} | -byBumpSite {list}}
```

Removes the Verilog signal names from bumps, which removes the connection between bumps and I/O pins. The original connection will remain in the Verilog and is connected when the bump is reassigned. If the bump is assigned, it will appear in the DEF SPECIALNETS section with the signal (net) name. Use this command after bumps are assigned with the assignBump command or defined in the I/O file with the signal name, but without the -fixed constraint.

Parameters

-allBumps	Removes all the connections between bumps and I/O pins and removes the signal assignment of the bumps.
-byBumpName *list*	Removes only the connections between bumps and I/O pins contained in the bump name list.
-byBumpSite *list*	Removes only the connections between bumps and I/O pins contained in the site list.

Examples

- The following command unassigns all of the bumps:

  ```
  unassignBump -allBumps
  ```

- The following command unassigns the (named) bumps bumpA and bumpB:

  ```
  unassignBump -byBumpName {bumpA bumpB}
  ```

unassignBumpByName

`unassignBumpByName` *`bump_name`*

Removes the signal assignment from a bump and deletes the `BUMP_ASSIGNMENT` property from the DEF file if the bump is unassigned.

Parameters

`bump_name`	Specifies the name of the bump for signal unassignment.

unfixBondPad

```
unfixBondPad
     {-ioInstName InstName [-pinName pinName] | -selected}
```

Unassigns the fixed status of a specified bond pad.

The following commands provide additional support for the staggering of bond pads:

- fixBondPad
- getBondPad
- placeBondPad
- staggerBondPad

Parameters

`-ioInstName` *`instName`*	Specifies the name of the I/O instance whose bond pad's fixed status will be unassigned.
`-pinName` *`pinName`*	Specifies the name of the pin.
`-selected`	Unassigns the fixed status of a selected bond pad. You must first select the bond pad in the design display area, then use this parameter to unassign the fixed status.

Example

- The following command unassigns the fixed status of the bond pad on I/O instance `pad_addr[30]`:

  ```
  unfixBondPad -ioInstName {pad_addr[30]}
  ```

unfixBump

```
unfixBump
     {-allBumps | -byBumpName {list}}
```

Removes the bump assignment status from a specified list of assigned bumps that were set with the setBumpFixed command. The software updates the DEF file by removing the +FIXEDBUMP attribute from the specified group of assigned bumps to indicate that they are now in an unfixed state.

Parameters

`-allBumps`	Removes the bump assignment status from all bumps.
`-byBumpName {list}`	Removes the bump assignment status from a specified list of assigned bumps.

Examples

- The following command removes the bump assignment status from all bumps:

  ```
  unfixBump -allBumps
  ```

- The following command removes the bump assignment status from bumps bumpA and bumpB:

  ```
  unfixBump -byBumpName {bumpA bumpB}
  ```

unsetProbePin

`unsetProbePin {-instName *InstName* -pinName *pinName* | -*bump bumpName*}`

Unsets a pad or bump as a probe point. Probe points are used for package-level design and testing. To set or report on probe points, use the setProbePin or reportProbePins commands respectively.

Parameters

`-bump` *`bumpName`*	Specifies the instance name of the bump.
`-instName` *`InstName`*	Specifies the name of the I/O instance.
`-pinName` *`pinName`*	Specifies the name of the pin on the I/O instance that will be unset as a probe point.

Example

- The following command unsets the PAD pin of Bump_1 as a probe point:

```
unsetProbePin -instName Bump_1 -pinName PAD
```

viewBumpConnection

```
viewBumpConnection
     [-help]
     [-remove]
     [-bumpType {all | power | signal}]
     [-target {pad | ring}]
     [-multiBumpsToPad]
     [-multiPadsToBump]
```

Specify this command to view the connection displayed as a flight line between a power/ground bump and a power/ground I/O pad, during signal bump assignment.

Note: To remove the flight lines between the bumps and I/O pads, use `viewBumpConnection -remove`.

Parameters

`-bumpType {all | power | signal}`

Specifies the connecting bump type, which could be:

- `all` - All the bump types
- `power` - Power bump
- `signal` - Signal bump

`-help` Outputs the command usage.

`-multiBumpsToPad`

Connects multiple bumps to one pad.

`-multiPadsToBump`

Connects multiple pads to one bump.

`-remove` Removes the bump connections.

`-target {pad | ring}`

Specifies the connecting target, which could be a `pad` or a `ring`.

7

Floorplan Commands

addHaloToBlock

```
addHaloToBlock
     left bottom right top
     [instName | -allMacro | -allBlackBox | -allCommitPtn
          | -allBlock | -cell cellName]
     [-ori value]
     [-fromInstBox]
```

Adds a halo to a block. A halo is an area that prevents the placement of blocks and standard cells within the specified halo distance from the edges of a hard macro, black box, or committed partition in order to reduce congestion. A block halo value is specified based on the current block orientation.

Use this command during or after floorplanning.

Parameters

`-allBlackBox`	Specifies halo values for all black boxes.
`-allBlock`	Specifies halo values for all hard macros, black boxes, and committed partitions.
`-allCommitPtn`	Specifies halo values for all committed partitions.
`-allMacro`	Specifies halo values for all hard macros.
`bottom`	Specifies the distance, in micrometers, from the bottom edge of the block to the end of the halo area.
`-cell` *`cellName`*	Specifies halo values for all instances of a cell.
`-fromInstBox`	Creates a block halo, based on the instance boundary, even when there is an overlap layer defined in the block cell or macro definition.  *Caution* ***This parameter is for backward compatibility only. Cadence does not recommend the use of this parameter because there can be problems with the defin flow if this parameter is specified and if the block cell or the macro has overlap layers.***
`instName`	Specifies the block where the halo is to be added.

left	Specifies the distance, in micrometers, from the left edge of the block to the end of the halo area.
-ori *value*	Specifies the orientation value of the instances to which the block halo values are applied. The orientation value can be R0, R90, R180, R270, MX, MX90, MY, or MY90.
right	Specifies the distance, in micrometers, from the right edge of the block to the end of the halo area.
top	Specifies the distance, in micrometers, from the top edge of the block to the end of the halo area.

Example

The following command adds a halo around block xy_inst that is 10 µm from the left and right edges, and 20 µm from the bottom and top edges:

```
addHaloToBlock 10 20 10 20 xy_inst
```

addInstToInstGroup

```
addInstToInstGroup
     groupName {hInstName | instName | groupName | inst_name_list}
```

Adds a hierarchical instance, an instance or list of instances, or a group to a specified group. A group is a user-defined pseudo hierarchy and is created to contain instances and/or other groups.

Note: When you add a member to a group whose constraint is a fence or a region, the size of the bounding box of the group does not change.

Use this command after creating a group name.

Parameters

groupName Specifies the name of the created group.

hInstName | *instName* | *groupName*

Specifies the name of the object.

groupName	Specifies the name of another group
hInstName	Specifies the name of the hierarchical instance.
instName	Specifies the name of the leaf instance.
inst_name_list	Specifies the list of instance names. **Note:** The names list option is for instance only and not for `group` or `hInst`.

Examples

- The following command adds hierarchical instance to group `adder1`:

  ```
  addInstToInstGroup adder1 SH28/I12/I44
  ```

- The following command adds group `subadder` to group `adder1`:

  ```
  addInstToInstGroup adder1 subadder
  ```

- The following command adds the instances `inst_I1`, `inst_I2`, and `inst_I3` to group `inst_group`:

  ```
  addInstToInstGroup inst_group {inst_I1 inst_I2 inst_I3}
  ```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Grouping Instances"

addIoFiller

```
addIoFiller
     -cell fillerCellName
     [-prefix prefix]
     [-side {n | w | s | e}]
     [-from coord]
     [-to coord]
     [-row rowNumber]
     [-fillAnyGap]
     [-useSmallIoHeight]
     [-fillerOrient R0 | R90 | R180 | R270 | MX | MX90 | MY | MY90]
```

Adds I/O instances in the I/O box. The I/O instances are added between the gap of existing I/O pad instances where the gap is large enough for the I/O instance.

Note: Before using `addIoFiller`, run the `globalNetConnect` command to provide global-net-connection rules for supply pins of the added fillers. Without these rules, the built-in design-rule checks of `addIoFiller` will not be accurate.

Use this command during or after floorplanning.

Parameters

`-cell fillerCellName`

Specifies the name of the I/O filler cell to add.

`-fillAnyGap`

Forces the I/O filler instance into a gap even though the gap (clearance) is not large enough.

Default: If you do not specify this parameter, it does not force a filler instance.

`-fillerOrient R0 | R90 | R180 | R270 | MX | MX90 | MY | MY90`

Specifies the orientation value of the filler cell. The orientation value can be `R0`, `R90`, `R180`, `R270`, `MX`, `MX90`, `MY`, or `MY90`.

`-from coord`

Specifies the coordinates of a specified starting range where I/O filler cells are added. The values are in micrometers.

`-prefix prefix`

Specifies the prefix name to be appended for the added I/O filler instance.

`-row rowNumber`

Specifies the row number of the I/O pads (in multiple ring I/O pads) to insert the fillers.

`-side {n | w | s | e}`

Specifies the side as North side, `w` as West side, `s` as South side, and `e` as East side of the I/O box.

Default: If you do not specify this parameter, all sides are specified.

`-to` *`coord`*

Specifies the coordinates of a specified ending range where I/O filler cells are added. The values are in micrometers.

`-useSmallIoHeight`

Specifies that the filler cell should be inserted between I/O pads only when the smallest I/O pad has the same height as the height of the filler cell.

For example, consider that there are I/O pads A, B, C, D, and E. Out of these, A, B, and E are the same height, and C and D have a height double that of A, B, and E. Assume that you want to insert an I/O filler cell F, which has the same height as A, B, and E.

If you do *not* specify the `-useSmallIoHeight` parameter, the cell F will be inserted between A and B, B and C, C and D, and D and E. However, if you specify the `-useSmallIoHeight` parameter, the cell F will be inserted between A and B, B and C, and D and E. However, the cell will not be inserted between C and D, because both C and D have a height different from F.

Note: The `-from` and `-to` parameters are intended to be used with the `-side` parameter to indicate a range. This command is applicable to peripheral I/O only; therefore, it is only one dimensional.

Examples

- The following command adds I/O instances to the I/O pad area on all sides wherever there is space between existing I/O pads. The added I/O pad filler instance names have a prefix PAD8.

  ```
  addIoFiller -cell PAD8 -prefix PAD8
  ```

- The following command adds I/O instances to the I/O pad area on the west side from 200 micrometers. The added I/O pad filler instance names have a prefix PAD8.

  ```
  addIoFiller -cell PAD8 -prefix PAD8 -side w -from 200
  ```

- The following command adds I/O instances to the I/O pad area on the north side to 300 micrometers. The added I/O pad filler instance names have a prefix PAD8.

```
addIoFiller -cell PAD8 -prefix PAD8 -side n -to 300
```

- The following command adds I/O filler instance `myIoFiller` to the power domains, `PD1` and `PD2`, on the north side to 300 micrometers. The added I/O pad filler instance names have a prefix `FILL_VDD1_PD1 and FILL_VDD2_PD2`, indicating that this is a multi-VDD design.

```
addIoFiller -cell myIoFiller -prefix FILL_VDD1_PD1 side n -to 300
addIoFiller -cell myIoFiller -prefix FILL_VDD2_PD2 side n -to 300
```

addIoInstance

```
addIoInstance
     [-help]
     [-cell {name1 name2 ...}]
     -inst {inst1 inst2 ...}
     {{-refInst inst_name [-ccw]} |
     {-selected [-ccw]} |
     {-repeat num [-skip num]
     [-inSelected | [-ioRing num] -side {N | W | S | E} |
     -ioRow name]} |
     -loc llx lly
     }
     [-spacing val]
     [-orientation orient]
     [-spread]
```

Specifies constraints for insertion of power and ground I/O instances in the I/O ring or in the I/O rows.

If you do not specify any of the following parameters, the command `addIoInstance` adds I/Os for all I/O cells, on all sides and for all I/O rings.

- `-ioRing` *num*
- `-side`
- `-ioRow` *name*

Parameters

`-ccw`	Specifies the counter-clockwise direction to add the new I/O instance.
`-cell {`*name1 name2 ...*`}`	Specifies one or more cells. Adds one I/O for each cell that was specified
`-help`	Outputs a brief description that includes type and default information for each `addIoInstance` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man addIoInstance`.
`-inSelected`	Repeatedly inserts new I/O instance(s) within the selected I/O cell(s).

`-inst {`*`inst1 inst2 ...`*`}`

Specifies the instance name prefix for the new I/O. This must match with the number of cells that are specified.

`-ioRing` *`num`*

Specifies the I/O ring number in which the new I/O instance is added.

`-ioRow` *`name`*

For the new I/O row flow, adds I/Os on the specified I/O row.

`-loc` *`llx lly`*

Adds the new I/O instance at the location specified by lower-left x (*`llx`*) and lower-left y (*`lly`*) coordinates.

Note: You cannot use this parameter in conjunction with the `-refInst` or `-selected` parameters.

`-orientation` *`orient`*

Specifies the orientation of the new I/O instance. You can specify one of the following values:

- `R0`
- `R90`
- `R180`
- `R270`
- `MX90`
- `MY90`
- `MX`
- `MY`

Default: `R0`

Use this parameter with `-cell` and `-loc` parameters.

Note: When specifying the `-refInst` or `-selected` parameters, the software uses the orientation from those instances.

See Orientation Key for more information.

`-refInst` *`inst_name`*

	Specifies the name of the reference instance and adds the new I/O instance above or below the reference instance in clockwise direction. *Default*: Clockwise direction of the reference instance.
`-repeat` *num*	Repeatedly adds new I/O instances every number (num) of cells that you specify. *Default*: Adds instances in the clockwise direction of the reference instance.
`-selected`	If `-repeat` is specified, this parameter adds the new I/O instance(s) in the selected cell(s) in clockwise direction of the selected instance. If `-repeat` is not specified, this parameter changes the selected cells as referenced cells and adds new I/O instances in the referenced cells, in the default clockwise direction.
`-side {N | W | S | E}`	Specifies the side on which the new I/O instance is added.
`-skip` *num*	Skips the number (num) of cells specified, before adding the first I/O instance. This happens only when `-repeat num` is specified.
`-spacing` *val*	Specifies the spacing value, in microns, between the I/O instances.
`-spread`	Evenly replaces all the I/O cells on the same row where the new I/O instance was added. This prevents overlapping of I/O cells. If you do not specify this parameter, the command still adds the new I/Os even if overlapping occurs. **Note:** The `addIoInstance` command works for both the I/O flows, the old flow and the new row based pad placement flow. In the new flow, each time you add a new I/O instance, the I/Os on the I/O rows with the existing constraints are replaced. In this situation, if all the existing I/Os do not have any specific constraint defined, then the replacement of the I/Os also resolves overlapping each time when the I/O cell is added, evading the usage of `-spread` option.

Example

- The following command adds new I/O instance `il` to cell `BUMPCELL` at lower-left x location `100` and lower-left y location `100`:

  ```
  addIoInstance -inst il -cell BUMPCELL -loc 100 100
  ```

- The following command inserts the I/O instance `newIo` for the cell `PVDD` on the West side of the I/O box, skipping 3 I/O's (from below) on the West side and repeating the insertion of I/O's every 2 instances.

  ```
  addIoInstance -cell PVDD -repeat 2 -inst newIo -side W -skip 3 -spread
  ```

addIoRowFiller

```
addIoRowFiller
     [-help]
     -cell {fillerCellNameList ...}
     [-prefix prefix]
     [-ioRow name [-fillerOrient R0 | R90 | R180 | R270 | MX | MX90 | MY | MY90]]
     [-from coord] [-to coord]
     [-fillAnyGap]
     [-useSmallIoHeight]
     [-ignoreSiteType]
```

Adds I/O filler cells in the I/O rows. The I/O filler cells are added between the gap of existing I/O pad instances where the gap is large enough for the I/O cell.

Use this command during or after floorplanning.

Parameters

`-cell{`*`fillerCellNameList ...`*`}`

Specifies the name of the filler cell(s) to add in the I/O row.

`-fillAnyGap`

Forces the I/O filler instance into a gap even though the gap (clearance) is not large enough.

Default: If you do not specify this parameter, it does not force a filler instance.

`-fillerOrient R0 | R90 | R180 | R270 | MX | MX90 | MY | MY90`

Specifies the orientation value of the filler cell. The orientation value can be `R0`, `R90`, `R180`, `R270`, `MX`, `MX90`, `MY`, or `MY90`.

`-from` *`coord`*

Specifies the coordinates of a specified starting range where I/O filler cells are added. The values are in micrometers.

`-help`

Outputs a brief description that includes type and default information for each `addIoRowFiller` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man addIoRowFiller`.

`-ignoreSiteType`

Ignores the I/O row site type.

`-ioRow` *`name`*

Specifies the name of the I/O row to add the filler cell.

`-prefix` *`prefix`*

Specifies the prefix name to be appended for the added I/O row filler cell.

`-to` *`coord`*

Specifies the coordinates of a specified ending range where I/O row filler cells are added. The values are in micrometers.

`-useSmallIoHeight`

Specifies that the filler cell should be inserted between I/O pads only when the smallest I/O pad has the same height as the height of the filler cell.

For example, consider that there are I/O pads A, B, C, D, and E. Out of these, A, B, and E are the same height, and C and D have a height double that of A, B, and E. Assume that you want to insert an I/O filler cell F, which has the same height as A, B, and E.

If you do *not* specify the `-useSmallIoHeight` parameter, the cell F will be inserted between A and B, B and C, C and D, and D and E. However, if you specify the `-useSmallIoHeight` parameter, the cell F will be inserted between A and B, B and C, and D and E. However, the cell will not be inserted between C and D, because both C and D have a height different from F.

addModuleToFPlan

`addModuleToFPlan` *`hInstName`*

Adds a hierarchical module instance to the floorplan view. If a module is too small in size, the module will not be displayed in the floorplan view.

Tip

You can change the *Min. floorplan Module Size* option in the Preferences form in the *Display* menu and import the design.

Use this command after importing the design.

Parameters

`hInstName`	Specifies the name of the hierarchical instance.

Example

The following command adds hierarchical instance `SH99` in the floorplan view:

```
addModuleToFPlan SH99
```

addObjFPlanCutBox

```
addObjFPlanCutBox
     objectType
     objectName
     llx1 lly1 urx1 ury1
     llx2 lly2 urx2 ury2
     ...
     llxn llyn urxn uryn
```

Specifies the rectilinear shape of floorplan objects by describing the cut out area. The floorplan objects include block top cell, module constraint (guide, region, fence), layershape, pin, net, group, and instance.

The command creates a rectilinear cut box for an object. A rectilinear object is comprised of two or more boxes.

To use this command, divide the required cut out area into one or more bounding boxes and specify the coordinates of each bounding box in the command. See the Example 7-1 on page 385.

Once you run the command, the rectilinear cut area disappears from the object after a redraw.

For specifying the rectilinear shape of objects by describing the device area, see setObjFPlanBoxList on page 573.

Parameters

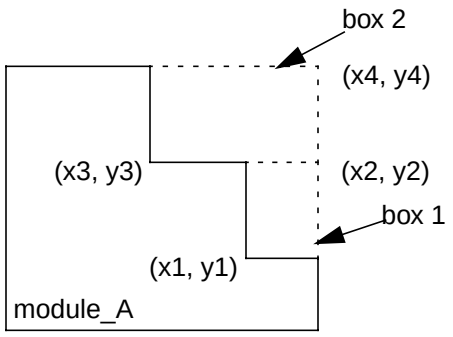
objectName	Specifies the name of the object for *objectType*.
objectType	Specifies the type of object for *objectName*. This can be: ■ Cell ■ Group ■ Instance ■ Layershape ■ Module ■ Net ■ Pin
llx1	Specifies the lower left x coordinate of the first cut area.
lly1	Specifies the lower left y coordinate of the first cut area.

llx2	Specifies the lower left x coordinate of the second box in the box list.
lly2	Specifies the lower left y coordinate of the second box in the box list.
urx1	Specifies the upper right x coordinate of the first cut area.
ury1	Specifies the upper right y coordinate of the first cut area.
urx2	Specifies the upper right x coordinate of the second cut area.
ury2	Specifies the upper right y coordinate of the second cut area.
llxn	Specifies the lower left x coordinate of the n^{th} cut area.
llyn	Specifies the lower left y coordinate of the n^{th} cut area.
urxn	Specifies the upper right x coordinate of the n^{th} cut area.
uryn	Specifies the upper right y coordinate of the n^{th} cut area.

Example 7-1

The following example defines the rectilinear shape of the module `module_A` by specifying 2 rectilinear cut boxes defined by coordinates (`x1`, `y1`), (`x2`, `y2`), (`x3`, `y3`), and (`x4`, `y4`).

```
addObjFPlanCutBox Module module_A x1 y1 x2 y2 x3 y3 x4 y4
```



addRoutingHalo

```
addRoutingHalo
     {-allBlocks | -block blockNameList | -inst instanceName | -designHalo}
     -space haloValue
     -top topLayer
     -bottom bottomLayer
```

Adds a routing halo for blackboxes, hard macros, or block-level designs. A routing halo significantly reduces the possibility of long wire routing within the specified area, thus helping reduce signal integrity violations at the top- and/or block-level designs.

A routing halo is honored only by the signal router. The signal router treats routing on the specific routing layers in the routing halo area as very high cost routing. However, perpendicular routing or straight connections to pins are acceptable. The special router does not honor routing halos. To block special routing, use routing blockages instead.

You can specify a routing halo for:

- hard macros or blackboxes
- block-level designs
- partitions

For the bottom-up hierarchical flow, use this command to specify routing halo for a block at the top-level design or at the block-level design.

Note: For top-down hierarchical flow, specify routing halo for a partition through the `definePartition` command or through the Specify Partition form.

The routing halo can be stored as an instance and/or design property and can be viewed through the attribute editor for the instance block. The routing halo can be saved to or restored from a floorplan file.

Use this command after importing a design.

Parameters

`-allBlocks` Specifies that the routing halo should be added to all blocks in the design.

`-block` *`blockNameList`*

	Specifies that the routing halo should be added to the specified blocks. A block name must be an instance name and not a cell master name. The specified blocks should be separated by a space.
`-bottom` *`bottomLayer`*	
	Specifies the bottom layer for the routing halo.
`-designHalo`	Specifies that the routing halo should be added at the block-level design.
`-inst` *`instanceName`*	Specifies that the routing halo should be added to the specified instance. This parameter can be useful for adding halo for I/O cells.
`-space` *`haloValue`*	Specifies the routing halo value in microns.
`-top` *`topLayer`*	Specifies the top layer for the routing halo.

Examples

- The following command adds a routing halo of 10 microns to all blocks and specifies that the top and the bottom layers for the routing halo are M7 and M1, respectively.

  ```
  addRoutingHalo -space 10 -top M7 -bottom M1 -allBlocks
  ```

- The following command adds a routing halo of 5 microns to the specified blocks and specifies that the top and the bottom layers for the routing halo are M2 and M1, respectively.

  ```
  addRoutingHalo -block DTMF_INST/RAM_256x16_TEST_INST/RAM_256x16_INST
  DTMF_INST/RAM_128x16_TEST_INST/RAM_128x16_INST -space 5 -top M2 -bottom M1
  ```

alignInst

```
alignInst
     -side {right | left | center | top | bottom | middle}]
     [-referToFirst]
```

Aligns instances vertically or horizontally.

Use this command after selecting two or more instances in the design display area.

Parameters

`-referToFirst`	Specifies a reference instance highlighted in the design display area. If you specify this option, the first selected instance is the reference object.	
`-side`	Specifies the vertical and horizontal instance alignment for the selected instances. Choose one of the following:	
	`right`	Specifies vertical right alignment.
	`left`	Specifies vertical left alignment.
	`center`	Specifies vertical center alignment.
	`top`	Specifies horizontal top alignment.
	`bottom`	Specifies horizontal bottom alignment.
	`middle`	Specifies horizontal middle alignment.

Example

The following command aligns highlighted instances `instA` and `instB` vertically to the left:

```
alignInst -side left
```

analyzeFloorplan

```
analyzeFloorplan
     [-help]
     [-cong]
     [-timing]
     [-effort {low | medium | high}]
     [-outfile fileName]
     [-tcl fileName]
     [-keepPlacement]
```

Analyzes the floorplan, and reports basic design information. To report wire length, congestion, and timing information in addition to basic design information, specify the `-cong` and `-timing` parameters.

The `analyzeFloorplan` command can be used with floorplans generated by Masterplan, generated manually, or generated by a third-party tool.

In order to use the `analyzeFloorplan` command, the following information must exist in the design:

- Verilog netlist
- Physical library (LEF) file
- Timing library (`.lib`) file
- SDC constraint (`.sdc`) file

Additionally, macros in the floorplan must be marked `FIXED` before running `analyzeFloorplan`.

Parameters

`-cong`
: Analyzes congestion and reports total wire length, and horizontal and vertical congestion information.

`-effort {low | medium | high}`
: Specifies the effort level to use for analysis. The `-effort` parameter allows you to choose between analysis accuracy and run time levels.

  Default: `medium`

	`low`	Provides cluster mode placement (non-timing driven). This results in low analysis accuracy, but a fast run time.
	`medium`	Provides timing driven floorplan mode placement. This results in medium analysis accuracy and run time.
	`high`	Provides timing driven default placement (`placeDesign`). This results in good analysis accuracy, but a longer run time.
`-help`	Outputs a brief description that includes type and default information for each `analyzeFloorplan` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man analyzeFloorplan`.	
`-keepPlacement`	Specify this option to retain the placement/trialRoute results after analyzing the floorplan.	
`-outfile` *`fileName`*	Saves the report to the specified output file. If you specify this parameter without giving *`filename`*, the software saves the report to a file named `analyzeFP.rpt`	
`-timing`	Analyzes timing and reports the total negative slack and worst negative slack for register-to-register paths.	

Examples

- The following command analyzes the floorplan and reports basic information on the design:

  ```
  analyzeFloorplan
  ```

  The software generates the following report:

  ```
  ##################################################################
  #  Generated by:      Cadence First Encounter 08.10-b005_1
  #  OS:                SunOS sun4u(Host ID pvsunray5)
  #  Generated on:      Mon Jun 23 18:08:08
  #  Command:           analyzeFloorplan
  ##################################################################
  ************************ Analyze Floorplan *************************
      Die Area(um^2)            : 1644650.00
  ```

```
    Core Area(um^2)           : 429571.30
    Number of instance(s)     : 7780
    Number of Macro(s)        : 4
    Number of IO Pin(s)       : 53
    Number of Power Domain(s) : 1
************************ Estimation Results ************************
********************************************************************
```

- The following command analyzes the floorplan and reports total wire length, and horizontal and vertical congestion information, in addition to the basic design information:

  ```
  analyzeFloorplan -cong
  ```

  The software generates the following report:

  ```
  ##################################################################
  #  Generated by:      Cadence First Encounter 08.10-b005_1
  #  OS:                SunOS sun4u(Host ID pvsunray5)
  #  Generated on:      Mon Jun 23 18:12:07
  #  Command:           analyzeFloorplan -cong
  ##################################################################
  ************************ Analyze Floorplan *************************
      Die Area(um^2)            : 1644650.00
      Core Area(um^2)           : 429571.30
      Number of instance(s)     : 7780
      Number of Macro(s)        : 4
      Number of IO Pin(s)       : 53
      Number of Power Domain(s) : 1
  ************************ Estimation Results ************************
      Total wire length.        : 3.4665e+05
      Congestion (H).           : 0.000%
      Congestion (V).           : 0.000%
  ********************************************************************
  ```

Related Topics

- "Creating An Initial Floorplan Using Masterplan" chapter of the *Encounter User Guide*
  - "Analyzing the Floorplan"

autoGenRelativeFPlan

```
autoGenRelativeFPlan
     [-fixedCell | -fixedPreroute]
     [-maxSpacing x]
     [-refCorner {BL|LB|BR|RB|TL|LT|TR|RT}]
```

Generates an initial pre-routed relative floorplan. You can use this command to generate a set of default pre-routed relative constraints. You can edit these generated pre-routed relative constraints with the `relativeFPlan --preRoute` command.

Note: If you do not specify the `-fixedCell` or the `-fixedPreroute` parameter, relative floorplan constraints will be generated for blocks and fixed standard cells as well as for fixed wire editing routes, that is, the routes created by wire editing

Use this command before running relative floorplanning.

Parameters

`-fixedCell`
: Specifies that relative floorplan constraints will be generated only for blocks and fixed standard cells.

`-fixedPreroute`
: Specifies that relative floorplan constraints will be generated only for fixed wire editing routes, that is, the routes created by wire editing.

`-maxSpacing` *x*
: Specifies the maximum space, in microns, between fixed cells and/or blocks for deriving constraints.

  Default: If you do not specify this option, the default value is 100.

`-refCorner {BL|LB|BR|RB|TL|LT|TR|RT}`
: Specifies the reference corner for cells. Choose one of the following:

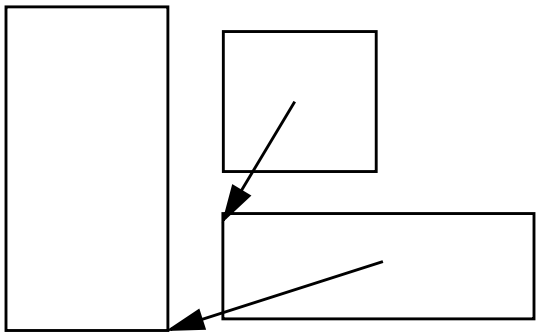
  `BL`
  : Bottom left. The position calculation relative to the cell will be first toward the bottom and then toward the left side.

LB

LB (left bottom). The position calculation relative to the cell will be first toward the left side and then toward the bottom.

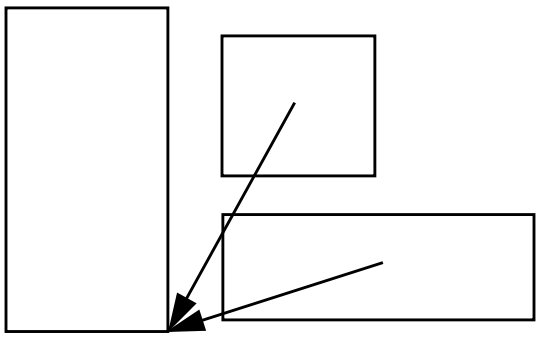
The difference between BL and LB is illustrated as follows:

Reference Corner BL



If you specify BL as the reference corner, the movement is first to bottom and then to the left.

Reference Corner LB



If you specify LB as the reference corner, the movement is first to left and then to the bottom.

There is a similar difference between BR and RB and between TL and LT.

BR	Bottom right. The position calculation relative to the cell will be first toward the bottom and then toward the right side.
RB	Right bottom. The position calculation relative to the cell will be first toward the bottom and then toward the right side.
TL	Top left. The position calculation relative to the cell will be first toward the top and then toward the left side.
LT	Left Top. The position calculation relative to the cell will be first toward the left side and then toward the top.
TR	Top right. The position calculation relative to the cell will be first toward the top and then toward the right side.
RT	Right top. The position calculation relative to the cell will be first toward the right side and then toward the top.

changeIoConstraints

```
changeIoConstraints
     [-help]
     [-ioOrder {clockwise | counterclockwise |default} | -row name]
     [-spacing num | -removeSpacing]
     [-endSpace num | -removeEndSpace]
     [-firstCellSpacing num | -removeFirstCellSpacing]
     [-removeAll]
```

Changes the constraints of an I/O row defined in the I/O constraint file. If you do not specify the name of the I/O row, the command modifies all the I/O rows.

Parameters

`-endSpace` *num*

Specifies the space, in μmeters, between the corner pad and the last I/O pad for the specified side of the row.

`-firstCellSpacing` *num*

Specifies the distance, in μmeters, from the first cell to the beginning of the I/O row, based on the I/O order.

`-help`

Outputs a brief description that includes type and default information for each `changeIoConstraints` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man changeIoConstraints`.

`-ioOrder {clockwise | counterclockwise |default}`

Changes the I/O placement order on all the rows.

Note: This parameter is ignored if you specify a row name.

`-removeAll`

Removes all the existing constraints on the I/O's. Evenly distributes all the I/O's in the rows.

`-removeEndSpace`

Removes the end space of the row.

`-removeFirstCellSpacing`

Removes the spacing of the first cell.

`-removeSpacing`

Remove the spacing constraint from I/O cells on a row.

`-row` *`name`*

Specifies the name of the row to modify the I/O constraints.

`-spacing` *`num`*

Specifies the global spacing, in μmeters, for all the I/O's on a row.

Note: `changeIoConstraints -ioOrder -spacing num` does not modify the spacing of corner I/O pads.

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Performing I/O Row Based Pad Placement"
- "Data Preparation" chapter of the *Encounter User Guide*
  - "Creating an I/O Assignment File"

checkFPlan

```
checkFPlan
     [-place]
     [-powerDomain]
     [-feedthrough]
     [-reportUtil]
     [-outFile fileName]
     [-ptnClone]
```

Checks the quality of the floorplan to detect potential problems before the design is passed on to other tools. This highlights the cells in the design display area with violation markers, where applicable.

Use this command after specifying the floorplan data or running an initial floorplan. Run `checkFPlan` on your final floorplan file.

Parameters

`-feedthrough`	Checks feedthrough buffer insertion.
`-outFile` *`fileName`*	Outputs detailed information for the specified checks. If you specify the `-place` option, the output of the `checkPlace` command is concatenated with this output file.
`-place`	Checks the placement.
`-ptnClone`	Checks that partition clones are aligned with their master on the power mesh. This checks that the master and clones partitions are inside a core boundary, and ensures that the partition fence instances sizes match the master instance.
`-powerDomain`	Checks the power domains.
`-reportUtil`	Reports target utilization (TU) and effective utilization (EU) for the entire design, fences, and regions (partitions, power domains, and regular fences).

checkMacroLLOnTrack

```
checkMacroLLOnTrack
     -useM2M3Track
```

By default, the command checks whether the lower left coordinates (`llx`, `lly`) of hard macros are at the intersection of `M1/M2` tracks, and reports error in Encounter if they are not. This is based on the assumption that `M1` and `M2` are both specified in opposite directions in a `65nm` process.

Parameters

`-useM2M3Track`	Checks the location of the macro against `M2/M3` tracks if both `M1` and `M2` are specified in the same horizontal direction in a `40nm` process.

clearRelativeFPlan

`clearRelativeFPlan`

Removes the relative floorplan information from the database.

Note: The arrows showing relative constraints are removed from the floorplan display.

Use this command after running `relativeFPlan`.

Parameters

None

convertFenceToLef

```
convertFenceToLef
     {hInstName | [partitionName]}
```

Converts a specified hierarchical instance or partition name to a LEF block.

Note: This command runs `commitPartition` and `savePartition` for the specified partition as if it was at the top-level. This will overwrite any partition or hierarchical instance that is specified in the `.lef` file in the top partitioned subdirectory. In addition, the `ptnname.def` file will overwrite the partition's `ptn.def` subdirectory file.

Use this command after floorplanning the design.

Parameters

hInstName	Specifies the hierarchical instance cell name.
partitionName	Specifies the partition name. If this is not specified, the command uses the hierarchical instance cell name as the partition name.

Example

The following command converts the `results_conv` hierarchical instance to a LEF block.

```
convertFenceToLef results_conv
```

createDensityArea

```
createDensityArea
     llx lly urx ury
     density
     -name screenName
```

Creates a density screen area. A density screen area is a floorplan object used to assign standard cell placement density in the design area.

The `createDensityArea` command can be used to create partial placement blockages even outside the core boundary.

Use this command after importing the design.

Parameters

density	Specifies the standard cell placement density percentage. Valid values are in increments of 5: 100, 95, 90, 85, 80, 75, 70, 65, 60, 55, 50, 45, 40, 35, 30, 25, 20, 15, 10, 5, 0. For example, setting a density percentage of 75 percent means that up to 75 percent of the defined density area will be used for placement. **Note:** If you enter any number that is not valid, the percentage defaults to the nearest valid value.
llx	Specifies the lower left x coordinate of the density area.
lly	Specifies the lower left y coordinate of the density area.
-name *screenName*	Specifies the name of the density screen area.
urx	Specifies the upper right x coordinate of the density area.
ury	Specifies the upper right y coordinate of the density area.

Note: After creating the density screen area, you can double-click on the object in the design display area to open the Attribute Editor to change the density value:

Hard	The area cannot be used to place blocks at any time during the session.
Soft	The area cannot be used to place blocks during standard cell placement, but can be used during in-place optimization, clock tree synthesis, or ECO Placement.

Partial Sets a percentage of the area that is unavailable for placement. Use the *Blockage Percentage* pull-down menu to select a percentage.

Example

The following command creates a density screen area with placement density of 50%:

```
createDensityArea 2374.1000 5673.2200 2893.0000 6189.6000 50
```

createFence

```
createFence
     hInstName | groupName
     llx lly urx ury
```

Creates a fence for a module or a group.

Use this command after importing the design.

Parameters

groupName	Specifies the name of the created group.
hInstName	Specifies the name of the hierarchical instance to be fenced.
llx	Specifies the lower left x coordinate of the fence area.
lly	Specifies the lower left y coordinate of the fence area.
urx	Specifies the upper right x coordinate of the fence area.
ury	Specifies the upper right y coordinate of the fence area.

Example

The following command places the module guide of SH17 in the floorplan view and changes its status to Fence:

```
createFence SH17 100 100 4898 4898
```

createGuide

```
createGuide
     hInstName | groupName
     llx lly urx ury
```

Creates a guide for a module or a group.

Use this command after importing the design.

Parameters

groupName	Specifies the name of the created group.
hInstName	Specifies the name of the hierarchical instance to be guided.
llx	Specifies the lower left x coordinate of the guide area.
lly	Specifies the lower left y coordinate of the guide area.
urx	Specifies the upper right x coordinate of the guide area.
ury	Specifies the upper right y coordinate of the guide area.

Examples

- The following command creates a guide for hierarchical instance `MEM_DSCAN`:
  ```
  createGuide MEM_DSCAN 3478.8000 9012.0000 6498.0000 143087.2000
  ```
- The following command creates a guide for group `adder1`:
  ```
  createGuide adder1 834.2000 1012.0000 2498.4000 3087.0000
  ```

createInstGroup

```
createInstGroup
     groupName
     [-guide | -region | -fence llx lly urx ury]
     [-density densityValue [-ar aspectRatioValue]]
     [-isPhyHier]
```

Creates a new instance group, even outside the core boundary.

Use this command after importing the design.

Parameters

`-density densityValue`

Specifies a placement density percentage. Use the decimal format, where 0.75 is for 75% density.

`-ar aspectRatioValue`

Specifies the aspect ratio. If this is not specified, 1 is the default.

groupName Specifies the name of the created group. A group is a user-defined group name and is created to contain hierarchical instances, instances (leaf instances), or other groups.

`-guide | -region |-fence llx lly urx ury`

Specifies the coordinates of the guide, region, or fence area.

- *llx* Specifies the lower left x coordinate of the guide area.
- *lly* Specifies the lower left y coordinate of the guide area.
- *urx* Specifies the upper right x coordinate of the guide area.
- *ury* Specifies the upper right y coordinate of the guide area.

`-isPhyHier` Specifies that the created group is a physical hierarchy. By default, the created group is not saved back to the netlist.

Examples

- The following command creates group `adder1`:

  ```
  createInstGroup adder1
  ```

- The following command creates group `gusher_4` and its placement guide at the above area location:

  ```
  createInstGroup gusher_4 -guide 2345.3 9962.1 4930.0 18244.8
  ```

- The following command creates group `adder2` with 80% of the area occupied by the instance area with an aspect ratio of 2:

  ```
  createInstGroup adder2 -density 8.0 -AR 2
  ```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Grouping Instances"

createIoRow

```
createIoRow
     [-help]
     -site site_name
     [{-side {N | W | S | E}
     [-rowMargin val]
     [-beginOffset val]
     [-endOffset val | -length len | -nrSites nr]
     | -corner {BL | BR | TR | TL} [-xOffset value] [-yOffset value]
     }
     [-orientation {R0 | R90 | R180 | R270}]
     [-name row_name]]
     | -deriveByCells
     | -deriveBySelection
```

Creates new I/O rows and edits the existing I/O rows.

Note: Before creating I/O rows for I/O row based pad placement, you need to set the I/O row flag using the `setIoFlowFlag` command.

Parameters

`-beginOffset val`

Specifies the starting location of the I/O row from the die edge.

For a horizontal row, this value is the lower X value.

For a vertical row, this value is the lower Y value.

Default: `0`, starting from the die edge.

`-corner {BL | BR | TR | TL}`

Specifies the location of the corner cell. This can be:

- `BL`: Bottom Left
- `BR`: Bottom Right
- `TR`: Top Right
- `TL`: Top Left

`-deriveByCells` Creates an I/O row for all the I/O cells in the design.

`-deriveBySelection`

Creates an I/O row only for the selected I/O cells.

Note: When you specify this parameter, some I/O rows may overlap depending on the I/O cell distribution.

`-endOffset` *`val`*

Specifies the end location of the I/O row from the die edge.

For a horizontal row, this value is the higher `X` value.

For a vertical row, this value is the higher `Y` value.

Default: `0`, ending from the die edge.

`-help`

Outputs a brief description that includes type and default information for each `createIoRow` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man createIoRow`.

`-length` *`len`*

Specifies the length of the row, in µmeters, starting from the die edge or `-beginOffset` *`val`*.

`-name` *`row_name`*

Specifies the name of the I/O row to be created.

`-nrSites` *`nr`*

Specifies the number of sites to define the length of the row.

Note: You can either specify `-endOffset` or `- length` or `-nrSites` each time you run the `createIoRow` command.

`-orientation {R0 | R90 | R180 | R270}`

Specifies the orientation of the created row.

If you do not specify this parameter, the default orientation for each side starting from the row on the south side is `R0`, rotate `R90` for the next side, and so on... as follows:

- `R0` - South West
- `R90` - South East
- `R180` - North East
- `R270` - North West

The default order of orientation is South, East, North, and West.

`-rowMargin` *`val`*

Specifies the distance, in μmeters, from the row edge to the die edge. The row edge is ‘top’ for a North side row, ‘bottom’ for a South side row, ‘left’ for a West side row, and ‘right’ for an East side row.

`-side {N | W | S | E}`

Specifies the side on which the row is to be created. If you do not specify the side, by default, one row is created on all sides (South, East, North, and West).

`-site` *`site_name`*

Specifies the name of the site defined in the LEF library.

`-xOffset` *`value`*

Specifies the distance, in μmeters, from the left or right edge of the corner row to the die edge.

`-yOffset` *`value`*

Specifies the distance, in μmeters, from the top or bottom edge of the corner row to the die edge.

Related Topics

- “Floorplan Menu” chapter of the *Encounter Menu Reference*
  - “Create I/O Row”
- “Floorplanning the Design” chapter of the *Encounter User Guide*

- "Performing I/O Row Based Pad Placement"

- "Data Preparation" chapter of the *Encounter User Guide*
  - "Creating an I/O Assignment Flow"

createNdr

`createNdr` *`ndrRuleName`*

Creates a non-default rule from a temporary structure.

Parameters

`ndrRuleName`	Specifies the name of the non-default rule that is to be created.

Example

```
createNdr testndr
```

Related Topics

- "Using the NanoRoute router" chapter in the *Encounter User Guide*
  - "Using Non-Default Rules"

createObsAroundInst

```
createObsAroundInst
     left bottom right top
     instName
```

Creates an obstruction around the specified instance.

Parameters

instName	Specifies the name of the instance around which you create the obstruction.
left bottom right top	Specifies halo values around the instance box.

createObstruct

```
createObstruct
     llx lly urx ury
     [-soft]
     [-name obstructName]
```

Creates a standard/soft cell placement blockage that can be placed even outside the core area. A placement blockage is a floorplan object used to block standard cell placements.

Use this command after importing the design.

Parameters

llx	Specifies the lower left x coordinate of the obstruction area.
lly	Specifies the lower left y coordinate of the obstruction area.
-name *obstructName*	Specifies the name of the placement blockage.
-soft	Specifies the soft blockage that is to be created.
urx	Specifies the upper right x coordinate of the obstruction area.
ury	Specifies the upper right y coordinate of the obstruction area.

Note: After creating the placement blockage, you can double-click on the object in the design display area to open the Attribute Editor and select from three blockage types:

Hard	The area cannot be used to place blocks at any time during the session. This is the default.
Soft	The area cannot be used to place blocks during standard cell placement, but can be used during in-place optimization, clock tree synthesis, or ECO Placement.
Partial	Sets a percentage of the area that is unavailable for placement. Use the *Blockage Percentage* pull-down menu to select a percentage. For example, a partial blockage of 75 percent means that only up to 25 percent of placement density is allowed in the area.

Example

The following command creates a soft blockage, `softBlockage2` in the design area.

```
createObstruct 3442.3600 3739.2000 3511.6500 5716.6300 -soft -name softBlockage2
```

Related Topics

- "Placing the Design" chapter of the *Encounter User Guide*
  - "Guiding Placement with Blockages"
- "Tools Menu" chapter of the *Encounter Menu Reference*
  - "Attribute Editor"

createPGPin

```
createPGPin
     [-geom layerId llx lly urx ury]
     [-net netName]
     pgPinName
```

Creates a power/ground pin as per the specified coordinates of the physical shape. You can use this command to create a power/ground pin without having to first create a power stripe with the addStripe command.

If the -geom parameter is not specified, only the logical pin is created. If the -geom parameter is specified, the physical pin is also created.

If a logical power/ground pin exists before this command is run, the net name will be ignored if it is specified.

If a logical power/ground pin does not exist before this command is run:

- If a net has been specified with the -net parameter, the power/ground pin will be created and attached to the net specified.
- If a net name has not been specified, the net name is assumed to be the same as the pin name. In this case, the power/ground pin will be created and attached to that nets.

Parameters

-geom *layerId llx lly urx ury*

Specifies the geometry of the physical pin.

layerId	Specifies the layer on which the power/ground pin will be created. The layer id should be a numerical number, ranging from 1 to 7.
llx	Specifies the lower-left x coordinate of the power/ground pin.
lly	Specifies the lower-left y coordinate of the power/ground pin.
urx	Specifies the upper-right x coordinate of the power/ground pin.
ury	Specifies the upper-right y coordinate of the power/ground pin.

`-net` *`netName`*	Specifies the name of the net to which the power/ground pin will be attached.
`pgPinName`	Specifies the name of the power/ground pin.

Examples

- This example is for the case where a logical pin name has *not* been defined prior to running the command and a net name has been specified.

  The following command creates a power/ground pin named `pgpin_A` on layer 2 and attaches the pin to the net `net_A`. The pin coordinates are 7 (llx), 3 (lly), 17 (urx), and 23 (ury).

  ```
  createPGPin -geom 2 7 3 17 23 -net net_A pgpin_A
  ```

- This example is for the case where a logical pin name has not been defined prior to running the command and a net name has *not* been specified.

  The following command creates a power/ground pin named `pgpin_B` on layer 3 and attaches the pin to the net `pgpin_B`. The pin coordinates are 7 (llx), 3 (lly), 17 (urx), and 23 (ury). As the net name was not specified, the net name is assumed to be the same as the pin name.

  ```
  createPGPin -geom 3 7 3 17 23 pgpin_B
  ```

- This example is for the case where a logical pin name has already been defined prior to running the command.

  The following command creates a power/ground pin named `pgpin_C` on layer 5. The pin coordinates are 7 (llx), 3 (lly), 17 (urx), and 23 (ury). In this case, the logical pin name was already defined prior to running the command, and so a net name is not required. Even if a net name were specified, it would have been ignored.

  ```
  createPGPin -geom 5 7 3 17 23 pgpin_C
  ```

- This example is for the case where the net name is not specified and the `-geom` parameter has not been specified.

  The following command creates a logical pin names `pgpin_D`. The net name is assumed to be the same as the pin name and no physical pin is created.

  ```
  createPGPin pgpin_D
  ```

Related Topics

- Power Menu chapter of the *Encounter Menu Reference*
  - Create P/G Pin

createRegion

```
createRegion
     {hInstName | groupName}
     llx lly urx ury
```

Creates a region for a module or a group.

Use this command after importing the design.

Parameters

groupName	Specifies the name of the created group. A group is a user-defined group name and is created to contain hierarchical instances, instances (leaf instance), or other groups.
hInstName	Specifies the name of the module instance.
llx	Specifies the lower left x coordinate of the fence area.
lly	Specifies the lower left y coordinate of the fence area.
urx	Specifies the upper right x coordinate of the fence area.
ury	Specifies the upper right y coordinate of the fence area.

Example

The following command creates a module guide in the floorplan view with status Region:

```
createRegion SH17 100 100 4900 4900
```

createRouteBlk

```
createRouteBlk
     {-box <llx> <lly> <urx> <ury> | -polygon x1 y1 x2 y2 ...| -cover} ...
     [-layer {layerId ... | all}]
     [-name layerBlkName]
     [-cutLayer {layerId ... | all}]
     [-fills | -exceptpgnet | -inst name]
     [-spacing value | -designRuleWidth value]
```

Creates a routing blockage object that can be moved even outside the core area. The object area prevents routing of specified metal layers, signal routes, and hierarchical instances in this area.

Use this command during partition floorplanning.

Parameters

`-box llx lly urx ury`

Specifies the coordinates of the blockage area.

llx	Specifies the lower left x coordinate of the blockage area.
lly	Specifies the lower left y coordinate of the blockage area.
urx	Specifies the upper right x coordinate of the blockage area.
ury	Specifies the upper right y coordinate of the blockage area.

`-cover`

Specifies that a routing blockage of the same size as the specified block instance (`-inst name`) will be created on top of the instance.

The `-inst` parameter must be specified with this parameter.

`-cutLayer {layerId ... | all}`

Specifies the cut layer, list of cut layers, or all cut layers on which the routing blockage is to be applied.

To specify cut layer between metal 1 and metal 2, use layerId 2, for example, `-cutLayer 2`. Similarly, if you want to create blockage on cut layer between metal 2 and metal 3, use layerId 3, for example, `-cutlayer 3`, and so on.

Note: The values specified with this parameter will override any default blockage layers specified with the `setRouteBlkDefaultLayer` command.

Default: If you do not specify this parameter, the Encounter software will check if any default blockage layers have been specified with the `setRouteBlkDefaultLayer` command. If yes, the layers specified with the `setRouteBlkDefaultLayer` command will be used. If no, the default is *wire3* if the number of layers is greater than 3, or *wire2* if the number of layers is less than or equal to 3.

`-designRuleWidth` Specifies the effective width of the routing blockage. The value is a real number greater than 0.

`-exceptpgnet` Specifies that the routing blockage is to be applied on a signal net routing and *not* on power or ground net routing.

Note: Blocking the signal net routing helps in avoiding cross talk or coupling caused by signal routes.

`-fills` Specifies that the routing blockage is to be applied on metal fills.

`-inst` *`name`* Specifies the name of the hierarchical instance on which the routing blockage is to be applied.

`-layer {`*`layerId ...`* `| all}`

	Specifies the layer, list of layers, or all layers on which the routing blockage is to be applied. **Note:** The values specified with this parameter will override any default blockage layers specified with the `setRouteBlkDefaultLayer` command. *Default:* If you do not specify this parameter, the Encounter software will check if any default blockage layers have been specified with the `setRouteBlkDefaultLayer` command. If yes, the layers specified with the `setRouteBlkDefaultLayer` command will be used. If no, the default is *wire3* if the number of layers is greater than 3, or *wire2* if the number of layers is less than or equal to 3.
`-name` *`layerBlkName`*	Specifies the layer name of the route blockage.
`-polygon` *`x1 y1 x2 y2 ...`*	Specifies the polygon vertices of the blockage, in microns.
`-spacing` *`value`*	Specifies the minimum spacing between layers within the routing blockage. The value is a real number greater than or equal to 0.

Example

- The following command creates a routing blockage object to block `metal3` routing in this area:

  ```
  createRouteBlk -box 2353.8000 7123.000 2653.8000 8298.2000 -layer 3
  ```

- The following command creates a routing blockage object on metal layers 1, 4, and 5:

  ```
  createRouteBlk -box 0 0 100 100 -layer 1 4 5
  ```

- The following command creates a routing blockage object on metal layers 1, 4 and blocks the signal net routing:

  ```
  createRouteBlk -box 10 18 1782 2300 -layer 1 4 -exceptpgnet
  ```

- The following command creates a routing blockage object having polygon vertices (`10`, `10`), (`10`, `15`), (`15`, `15`), (`15`, `20`), (`20`, `20`), (`20`, `10`) on the metal layer `1`:

  ```
  createRouteBlk -polygon 10 10 10 15 15 15 15 20 20 20 20 10 -layer 1
  ```

createRow

```
createRow
     -site siteName
     [-area llx lly urx ury]
     [-spacing distance]
     [-noAbut | -noAbut1st]
     [-flip1st]
     [-noFlip]
```

Creates rows for the specified site. The row boundary can be defined by core area or the area that you specify. This command supports the creation of overlapping rows. This command can create only horizontal rows. By default, the rows are flipped and abutted.

The following points apply to the usage of this command:

- Rows can be created both inside and outside the core area, but within the die.
- All new rows created in an area must be an integer multiple of any existing rows in the same area.
- Non-integer multiple height rows cannot be created outside power domains.

Use this command after importing a design.

Parameters

`-area` *`llx lly urx ury`*

Specifies the coordinates of the area in which rows will be created. If you do not specify an area, the core area is taken by default.

`llx`	Specifies the lower left x coordinate of the area.
`lly`	Specifies the lower left y coordinate of the area.
`urx`	Specifies the upper right x coordinate of the area.
`ury`	Specifies the upper right y coordinate of the area.

`-flip1st` If the rows are flipped, specifies that the first row is flipped mirrored to its x-axis.

`-noAbut`	Specifies that rows should not be abutted (created back to back). *Default*: The rows are abutted by default.
`-noAbut1st`	If the rows are abutted, specifies that the first row should not be abutted.
`-noFlip`	Specifies that the rows should not be flipped. *Default*: The rows are flipped by default.
`-spacing` *`distance`*	Specifies the spacing between rows.
`-site` *`siteName`*	Specifies the name of the site to be used for creating the rows.

Example

- The following command creates rows for the site `lt3site` within the area defined by the coordinates 376.7, 577.57, 688.94, 725.68, maintaining a spacing of 1 micron between the rows, without the first row being abutted:

```
createRow -site lt3site -area 376.7 577.57 688.94 725.68 -spacing 1.0 -
noAbut1st
```

createSoftGuide

`createSoftGuide` *`hInstName`* | *`groupName`*

Creates a soft guide for a module or a group.

Use this command after importing a design.

Parameters

`groupName`	Specifies the name of the created group.
`hInstName`	Specifies the name of the hierarchical instance to be guided.

Examples

- The following command creates a soft guide for hierarchical instance `sub_sub_module2`:

  ```
  createSoftGuide top/sub_module1/sub_sub_module2
  ```

- The following command creates a soft guide for group `adder1`:

  ```
  createSoftGuide adder1
  ```

cutRectilinearInst

```
cutRectilinearInst -inst instName
     {-refInst instName [-spacingX xvalue] [-spacingY yvalue]} |
     {-cutBox {llx lly urx ury}} |
     {-corner cornerValue [-x value -y value]}
     [-pushOut sideEdgeName]
```

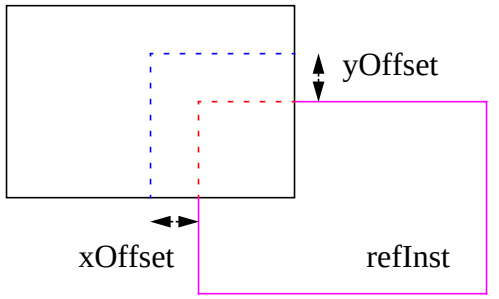
Cuts partition or black box boundaries to rectilinear instances that fit blocks in a given die, without modifying the existing block area. The command performs the rectilinear cut based on the following criteria that you specify:

- The name of a hierarchical instance.
- The name of a referenced instance.
- The name of corner where the cut is to be defined.
- The x and y offset values, in microns, from a specific corner or an overlapping area.
- The name of a side or edge that is to be pushed out, to maintain the existing block area.

Parameters

`-inst` *instName*	Specifies the name of a hierarchical instance. The instance can be a module, an uncommitted partition or a black box.
`-refInst` *instName*	Specifies the name of a referenced instance. The instance can be a module, an uncommitted partition, a black box or a hard macro.

`-spacingX` *`xValue`*	Specifies the x offset value, in microns, of the overlapping area between the specified block and the reference object. *`Default`*: 0

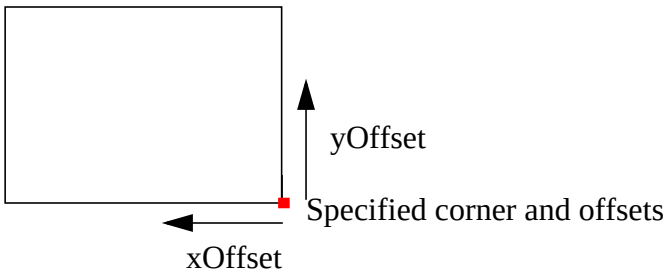


Reference instance with specified offsets

`-spacingY` *`yValue`*	Specifies the y offset value, in microns, of the overlapping area between the specified block and the reference object. *`Default`*: 0
`-cutBox`	Specifies the rectilinear box that is to be cut.
`llx`	Specifies the lower left x coordinate of the cut area.
`lly`	Specifies the lower left y coordinate of the cut area.
`urx`	Specifies the upper right x coordinate of the cut area.
`ury`	Specifies the upper right y coordinate of the cut area.
`-corner` *`cornerValue`*	Specifies the name of a corner from where the cut is to be defined. This can be lx, ly, ux, uy coordinates, or edge number in case of a rectilinear corner.

-x *xValue*

Specifies the x offset value, in microns, from a specified corner.



-y *yValue*

Specifies the y offset value, in microns, from a specified corner.

-pushOut sideEdgeName

Specifies the name of a side or edge that is to be pushed out, to maintain the existing area. This can be:

- top
- bottom
- left
- right
- rectilinear edge number

Note: If you do not specify a side or edge, the `cutRectilinearInst` command will not maintain the existing area and all the sides or edges are modified by default.

cutRow

```
cutRow
     [-area x1 y1 x2 y2 | -object objId | -selected]
     [-halo num | {-leftGap num | -rightGap num | -topGap num | -bottomGap num}]
     [-site siteName]
     [-keepCell]
```

Cuts site rows that intersect with the specified area or object.

Note: If no options are specified, the `cutRow` command automatically cuts all blocks and all rows around the placement blockage.

Use this command after importing a design.

Parameters

`-area` *x1 y1 x2 y2*

Specifies the x and y coordinates of the area in which rows will be created.

`-halo` *num*

Specifies the additional space to be provided on the top, bottom, left, and right sides of the specified or selected object. The halo (space) value is derived from the object boundary. The same value will be used for all sides.

`-keepCell`

Specifies that all cells that are placed inside the cut row area will not be unplaced.

Default: By default, all cells inside the cut row area will be unplaced.

`-leftGap` *num*
`-rightGap` *num*
`-topGap` *num*
`-bottomGap` *num*

Specifies the additional gap to be provided on the left, right, top, and bottom sides respectively of the specified or selected object.

`-object` *objId*

Specifies the name of the intersecting object.

`-selected`

Specifies that the rows that interfere with the selected object(s) will be cut.

`-site` *siteName*

Specifies the name of the site for which rows will be cut.

Default: Rows are cut for all core sites by default.

Example

```
cutRow -site SITE_FE283001 -area 124.975 138.123 237.378 223.38
```

deleteAllDensityAreas

```
deleteAllDensityAreas
```

Removes all partial placement blockages from the floorplan.

Use this command after importing the design.

Parameters

None

deleteAllFPObjects

`deleteAllFPObjects`

Removes all floorplan objects. The density screen, obstruction, power strips, and route guides objects are deleted. The guides and block guides are cleared from the design area. The physical instance groups are also cleared from the core area.

Note: To remove bus guides, the `deleteAllFPObjects` command internally calls the `selectBusGuide` command and the `deleteSelectedFromFPlan` command. For more information, see Clear Floorplan in the “Floorplan Menu” of the *Encounter Menu Reference*.

Use this command after importing the design.

Parameters

None

deleteAllInstGroups

```
deleteAllInstGroups
```

Removes all instance groups. Instance groups are user-created and contain instances and/or other groups.

Use this command after importing the design.

Parameters

None

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Grouping Instances"

deleteAllNetGroups

```
deleteAllNetGroups
```

Deletes all net groups that were created. A net group can contain net names, bus names, or segments of a bus.

Use this command after creating net groups.

Parameters

None

deleteAllPowerPreroutes

```
deleteAllPowerPreroutes
```

Removes all power preroutes from the floorplan.

Use this command after importing the design.

Parameters

None

deleteAllRouteBlks

```
deleteAllRouteBlks
```

Removes all route blockage objects in the floorplan.

Parameters

None

deleteAllSignalPreroutes

```
deleteAllSignalPreroutes
```

Removes all signal preroutes from the floorplan.

Use this command after importing the design.

Parameters

None

deleteFPObject

```
deleteFPObject
     objectType
     objectName
```

Deletes the specified floorplan object by name.

Parameters

objectName	Specifies the name of the object for the *ObjectType*.
objectType	Specifies the type of object for the *ObjectName*. The object type can be any of the following: ■ Bump ■ Cell ■ Group ■ Instance ■ I/O cell ■ I/O pin ■ Layer shape ■ Module ■ Net ■ Partition cut ■ Partition feedthrough ■ Partition pin block ■ Pin ■ Pin guide ■ Row cluster ■ Standard row

Example

The following command deletes the module abc from the current design:

```
deleteFPObject Module abc
```

deleteHaloFromBlock

```
deleteHaloFromBlock
     [instName | -allMacro | -allBlackBox | -allCommitPtn
          | -allBlock |
          -cell cellName]
```

Removes block halo values for specific blocks or for all blocks.

Use this command after running addHaloToBlock.

Parameters

`-allBlackBox`	Specifies the removal of halo values for all black boxes.
`-allBlock`	Specifies the removal of halo values for all blocks.
`-allCommitPtn`	Specifies the removal of halo values for all committed partitions.
`-allMacro`	Specifies the removal of halo values for all hard macros.
`-cell cellName`	Specifies the removal of halo values for all instances of a cell.
instName	Specifies the block where the halo is to be deleted.

Example

The following command removes block halo values from a hard macro named cube32:

```
deleteHaloFromBlock cube32
```

deleteInstFromInstGroup

```
deleteInstFromInstGroup
     groupName {hInstName | instName | groupName}
```

Removes a hierarchical instance, an instance, or a group from a specified group.

Use this command after importing the design.

Parameters

groupName Specifies the name of the created group.

hInstName | *instName* | *groupName*

Specifies the name of the object.

- *groupName* Specifies the name of another group. You can use wildcards (*?) with this parameter.
- *hInstName* Specifies the name of the hierarchical instance name. You can use wildcards (*?) with this parameter.
- *instName* Specifies the name of the leaf instance name. You can use wildcards (*?) with this parameter.

Examples

- The following command removes hierarchical instance `MEM_DSCAN` from group `adder1`:

  ```
  deleteInstFromInstGroup adder1 MEM_DSCAN
  ```

- The following command removes group `subadder` from group `adder1`:

  ```
  deleteInstFromInstGroup adder1 subadder
  ```

- The following command removes all instance names that begin with the letter `G` from group `adder1`:

  ```
  deleteInstFromInstGroup adder1 G*
  ```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Grouping Instances"

deleteInstGroup

`deleteInstGroup` *`groupName`*

Removes a group that was created.

Use this command after creating an instance group name.

Parameters

`groupName`	Specifies the name of the user created instance group name.

Example

The following commands create and remove the instance group `ADDER1`:

```
createInstGroup ADDER1
deleteInstGroup ADDER1
```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Grouping Instances"

deleteIoFiller

```
deleteIoFiller
     -cell fillerCellName
     [-prefix prefix]
     [-side {n | w | s | e}]
     [-from coord]
     [-to coord]
```

Deletes I/O filler cell instances from the design.

Use this command after using the addIoFiller command.

Parameters

`-cell` *`fillerCellName`*

Specifies the name of the I/O filler cell to remove.

`-from` *`coord`*
Specifies the coordinates of a specified starting range where I/O filler cells are to be removed. The values are in micrometers.

`-prefix` *`prefix`*
Specifies the prefix name of the I/O filler instance to be removed.

`-side {n | w | s | e}`

Specifies the side as North side, w as West side, s as South side, and e as East side of the I/O box where the I/O filler cells will be removed.

Default: If you do not specify this parameter, all sides are specified.

`-to` *`coord`*
Specifies the coordinates of a specified ending range where I/O filler cells are to be removed. The values are in micrometers.

deleteIoInstance

```
deleteIoInstance
     [-help]
     -instName {name1 name2 ...}
```

Deletes the specified I/O instance.

Parameters

`-help`	Outputs the command usage.
`-instName` *`list`*	Specifies the name(s) of I/O instance(s) to be deleted.

deleteIoRowFiller

```
deleteIoRowFiller
     -cell {fillerCellNameList ...}
     [-prefix prefix]
     [-ioRow name]
     [-from coord] [-to coord]
```

Deletes the I/O row filler cells.

Parameters

`-cell{`*`fillerCellNameList ...`*`}`

Specifies the name of the filler cell(s) to delete from the I/O row.

`-from` *`coord`*

Specifies the coordinates of a specified starting range where I/O filler cells are to be deleted. The values are in micrometers.

`-ioRow` *`name`*

Specifies the name of the I/O row to delete the filler cell.

`-prefix` *`prefix`*

Specifies the prefix name to be appended to the I/O row filler cell to be deleted.

`-to` *`coord`*

Specifies the coordinates of a specified ending range where I/O row filler cells to be deleted. The values are in micrometers.

deleteNetWeight

```
deleteNetWeight
     { -all | netName ... }
```

Removes the net weight values on the specified nets.

Use this command after running the `setNetWeight` command.

Parameters

`-all`	Specifies the removal of net weight values on all nets.
`netName`	Specifies the net names, separated by spaces, for which the net weight values are removed.

deleteObstruction

```
deleteObstruction {-all | obsName ...}
```

Removes all or individual placement obstructions from the floorplan.

Use this command after importing the design.

Parameters

`-all`	Specifies that all placement obstructions are removed from the floorplan. This is the equivalent of `deleteAllObstructs`.
`obsName ...`	Specifies the name of an obstruction or a group of obstructions, with each obstruction name separated by a space.

Example

The following command deletes obstructions `CTSOBS1` and `ABCOBS`:

```
deleteObstruction CTSOBS1 ABCOBS
```

deleteRelativeFPlan

```
deleteRelativeFPlan
     [objName]
```

Deletes relative constraints for the specified object, or removes the last executed relative floorplan action from the database.

Note: The arrows showing relative constraints for the specified objects are removed from the floorplan display.

Use this command after running `relativeFPlan`.

Parameters

objName	Deletes all relative constraints for the specified object(s). If you do not specify *objName*, this command removes the last executed relative floorplan action from the database.

deleteRouteBlk

```
deleteRouteBlk
     -box llx lly urx ury | -name blkName
     [-layer layerId ... | all]
     [-cutLayer {layerId ... | all}]
```

Deletes a routing blockage object.

Use this command during partition floorplanning.

Parameters

`-box llx lly urx ury`

Specifies the coordinates of the bounding box of the blockage area.

llx	Specifies the lower left x coordinate of the blockage area.
lly	Specifies the lower left y coordinate of the blockage area.
urx	Specifies the upper right x coordinate of the blockage area.
ury	Specifies the upper right y coordinate of the blockage area.

`-cutlayer {layerId... | all}`

Specifies the cut layer, list of cut layers, or all cut layers on which the routing blockage is to be deleted.

Default: If you do not specify this parameter, it deletes the routing blockage object on all cut layers.

`-layer {layerId... | all}`

Specifies the layer, list of layers, or all layers on which the routing blockage is to be deleted.

Default: If you do not specify this parameter, it deletes the routing blockage object on all layers.

`-name blkName` Specifies the layer name of the route blockage.

Examples

- The following commands create a blockage on all metal layers and deletes only on *metal3*:

```
createRouteBlk -box 11.2 164.07 95.005 319.205 -layer all
deleteRouteBlk -box 11.2 164.07 95.005 319.205 -layer 3
```

- The following commands create a blockage on all metal layers with the specific name BLK and deletes it.

```
createRouteBlk -box 11.2 164.07 95.005 319.205 -layer all -name BLK
deleteRouteBlk -name BLK
```

- The following commands create a blockage on all metal layers and deletes it.

```
createRouteBlk -box 400 400 500 500 -layer all
deleteRouteBlk -box 400 400 500 500
```

- The following commands create a blockage on *metal3* and deletes it.

```
createRouteBlk -box 400 400 500 500 -layer 3
deleteRouteBlk -box 400 400 500 500
```

deleteRoutingHalo

```
deleteRoutingHalo
     {-allBlocks | -block blockNameList | -inst instanceName | -designHalo}
```

Deletes a routing halo for a blackbox, hard macro, or block-level design.

Use this command after specifying a routing halo for a blackbox, hard macro, or block-level design.

Parameters

`-allBlocks`	Deletes the routing halo for all blocks and black boxes.
`-block` *`blockNameList`*	Deletes the routing halo for the specified blocks.
`-designHalo`	Deletes the routing halo at the design level.
`-inst` *`instanceName`*	Specifies that the routing halo should be added to the specified instance. Use this option to delete the halo of an I/O pad instance.

Example

```
deleteRoutingHalo -allBlocks
```

deleteRow

```
deleteRow
     {-all | -selected | -site siteName | rowPointer... | rowName...}
```

Use this command after importing a design.

Parameters

`-all`	Specifies that all rows will be deleted.
`rowName...`	Specifies the name of the row(s) to be deleted.
`rowPointer...`	Specifies the pointer to the row(s) to be deleted.
`-selected`	Specifies that the selected rows should be deleted.
`-site` *`siteName`*	Specifies the site for which rows will be deleted.

Example

The following command deletes rows for the site `site0A10.`

```
deleteRow -site site0A10
```

deleteSelectedFromFPlan

`deleteSelectedFromFPlan`

Deletes the currently selected floorplan objects.

Parameters

None.

deselectAll

```
deselectAll
```

Deselects all selected nets. Once the nets are deselected, they are no longer highlighted in the design display window.

Use this command after selecting one or more nets.

deselectGroup

`deselectGroup` *`groupName`*

Deselects a group.

Use this command after selecting a group.

Parameters

`groupName`	Specifies the name of the group.

deselectInst

`deselectInst` *`instanceName`*

Deselects the specified instance.

Use this command after selecting an instance.

Parameters

`instanceName`	Specifies the name of the instance.

deselectInstByCellName

`deselectInstByCellName` *`cellName`*

Deselects an instance by cell name.

Use this command after selecting an instance by cell name.

Parameters

`cellName`	Specifies the name of the cell.

deselectInstOnNet

`deselectInstOnNet` *`netName`*

Deselects an instance on a net.

Use this command after selecting an instance on a net.

Parameters

`netName`	Specifies the name of the net.

deselectIOPin

`deselectIOPin` *`pinName`*

Deselects an I/O pin.

Use this command after selecting an I/O pin.

Parameters

`pinName`	Specifies the name of the I/O pin.

deselectNet

```
deselectNet
     {netName | -clock | -nonDefaultRule | -shield}
```

Deselects the specified net. Once a net is deselected, it is no longer highlighted in the design display window.

Run this command after selecting one or more nets.

Parameters

netName	Specifies the name of the net you want to deselect. You cannot specify more than one net. You can, however, use a wild card (*).
-clock	Deselects all nets that have a DEF attribute + USE CLOCK
-nonDefaultRule	Deselects all nets that have a DEF attribute + NONDEFAULTRULE
-shield	Deselects all nets that have a DEF attribute + SHIELDNETS

Examples

- The following command deselects the net named net_1:

  ```
  deselectNet net_1
  ```

- The following command deselects all clock nets:

  ```
  deselectNet -clock
  ```

elaborateBlackBlob

`elaborateBlackBlob`

Instantiates the hard macros in a blackblob and completes the virtual connections for these hard macros. The blackblobs are also filled with dummy instances as per the size/area specified.

Typically, you would run the `elaborateBlackBlob` command once after you have specified the blackblobs in the design with the `specifyBlackBlob` command. For example, you might call the `specifyBlackblob` command several times in a script, and when all blackblobs have been specified, you would run the `elaborateBlackBlob` command once to instantiate the hard macros and complete the virtual connections for them.

The `elaborateBlackBlob` command is run automatically when you restore a design that contains blackblobs.

If you specify blackblobs through the Specify Blackblob GUI form, the `elaborateBlackBlob` command will automatically be invoked when you click the Apply or the OK button.

Parameters

None.

exportNdr

```
exportNdr ndrRuleName
     [-def defFileName]
     [-lef lefFileName]
```

Exports/saves the new rule that was created with the `createNdr` command, into a LEF/DEF file.

Parameters

ndrRuleName	Specifies the name of a non-default rule.
-def *defFileName*	Specifies the name of a DEF file, to which the non-default rule is exported.
-lef *lefFileName*	Specifies the name of a LEF file, to which the non-default rule is exported.

Example

The following command exports the non-default rule, `testndr` to the LEF file, `all.lef`.

```
exportNdr testndr -lef all.lef
```

Related Topics

- "Using the NanoRoute router" chapter in the *Encounter User Guide*
  - "Using Non-Default Rules"

finishFloorplan

```
finishFloorplan
     [-autoHalo | -addHalo width | -undoHalo]
     [-autoBlockage | -fillBlockage blkgType maxGap | -undoBlockage]
     [-staircase | -undoStaircase]
```

Performs advanced placement-related refinements to a floorplan, to produce a more polished floorplan.

The `finishFloorplan` command (with the exception of the `-staircase` parameter), can be used on any floorplan, including third-party generated floorplans.

Parameters

`-addHalo` *`width`*
: Adds placement halos of the specified width to every macro edge.

  Default: `-autoHalo`

`-autoBlockage`
: Adds partial (65 percent) placement blockages around macros to reduce potential congestion after placement.

`-autoHalo`
: Automatically add halos around every macro in design. For each macro edge, the halo thickness is proportional to pin density on that edge.

`-fillBlockage` *`blkgType maxGap`*
: Adds placement blockages of the specified blockage type (*`blkgType`*) to fill macro-to-macro and macro-to-boundary gaps up to a specified size (*`maxGap`*). For *`blkgType`*, you can specify `soft`, `hard`, or `partial`. For *`maxGap`*, specify a positive value, in microns.

  Default: `-autoBlockage`

`-staircase`
: Adjusts the macro sequence by size to smooth out the macro-to-core interface, creating a “staircase” pattern.

  You can specify `finishFloorplan -staircase` only on floorplans that were created using the `planDesign` command in the current session. Macros must be marked as `PLACED`, in order to specify this parameter.

`-undoBlockage`	Reverses the previous `-autoBlockage` or `-fillBlockage` operation. When specified, the software restores the floorplan to the state it was in before the last `finishFloorplan -autoBlockage` or `-fillBlockage` was specified. Any other refinement operations that were performed remain the same.
`-undoHalo`	Reverses the previous `-autoHalo` or `-addHalo` operation. When specified, the software restores the floorplan to the state it was in before the last `finishFloorplan -autoHalo` or `-addHalo` was specified. Any other refinement operations that were performed remain the same.
`-undoStaircase`	Reverses the previous `-staircase` operation. When specified, the software restores the floorplan to the state it was in before the last `finishFloorplan -staircase` was specified. Any other refinement operations that were performed remain the same.

fixAllIos

```
fixAllIos [-pinOnly | -cellOnly | -incAreaIo]
```

Changes the status of all I/O pins, I/O cells, or CLASS PAD AREAIO cells to a FIXED state to keep them from being reassigned.

Use this command after importing the design.

Parameters

-cellOnly	Changes the status of all I/O cells to a FIXED state.
-incAreaIo	Changes the status of all CLASS PAD AREAIO cells to a FIXED state.
-pinOnly	Changes the status of all I/O pins to a FIXED state.

Examples

- The following command changes the status of all I/O pins and I/O cells to a FIXED state:

  ```
  fixAllIos
  ```

  Note: This is the default functionality.

- The following command changes the status of all I/O pins to a FIXED state:

  ```
  fixAllIos -pinOnly
  ```

- The following command changes the status of all I/O cells to a FIXED state:

  ```
  fixAllIos -cellOnly
  ```

- The following command changes the status of all I/O pins, I/O cells, and all CLASS PAD AREAIO cells to a FIXED state:

  ```
  fixAllIos -incAreaIo
  ```

flipGroup

```
flipGroup
     groupName
     X | Y
     "location"
```

Flips all instances, with respect to the referenced X axis or Y axis at the location specified in the group.

Parameters

groupName	Specifies the name of the group whose instances are to be flipped.
location	Specifies the coordinates, in micrometers, in the group as the reference of X or Y axis to flip.
*X	Y*	Specifies the X or Y axis of the group to be used as a reference to flip all instances.

Example

The following command flips all instances in the group `abc` from the right side of the `Y` axis at location `"20 40"` to the left and from the left side to the right, and flips all instances mirrored through it's own Y axis as well:

```
flipGroup abc Y "20 40"
```

flipInst

```
flipInst
     inst
     {MX | MY}
```

Flips an instance (block) mirrored through its x or y axis.

If you flip a master instance blackbox to non-R0 orientation, the new non-R0 orientation will automatically be converted to R0 orientation. For more information, see Handling of Blackboxes with Non-R0 Orientation in the “Partitioning the Design” chapter of the *Encounter User Guide*.

Use this command after importing the design.

Parameters

inst	Specifies the name of the hierarchical instance to be flipped.
MX	Flips the instance mirrored through its x axis.
MY	Flips the instance mirrored through its y axis.

Example

The following command flips the block SH17/I441 mirrored through its x axis:

```
flipInst SH17/I4421 MX
```

flipModule

```
flipModule
     moduleName
     X | Y
```

Flips all hierarchical instances in the specified module, with respect to the referenced X axis or Y axis of the module.

Parameters

moduleName	Specifies the name of the module whose hierarchical instances are to be flipped.
*X	Y*	Specifies the X or Y axis of the module to be used as a reference to flip all hierarchical instances. This referenced X or Y axis is normally at the center of the module.

Example

The following command flips all hierarchical instances in the module abc from the right side of the referenced Y axis to the left and from the left side to the right, and flips all instances mirrored through it's own Y axis as well.

```
flipModule abc Y
```

floorPlan

```
floorPlan
     {-r aspectRatio [rowDensity
          [coreToLeft coreToBottom coreToRight coreToTop]]
     | -su aspectRatio [stdCellDensity [coreToLeft coreToBottom coreToRight
     coreToTop]]
     | -s coreW coreH
          coreToLeft coreToBottom coreToRight coreToTop
     | -d dieW dieH
          coreToLeft coreToBottom coreToRight coreToTop
     | -b die_x1 die_y1 die_x2 die_y2
          io_x1 io_y1 io_x2 io_y2
          core_x1 core_y1 core_x2 core_y2}
     [-overlapSameSiteRow]
     [-site siteName]
     [-flip {f | s | n}]
     [-fplanOrigin {center | lcorner}]
     [-dieSizeByIoHeight {max | min}]
     [-coreMarginsBy {io | die}]
     [-verticalRow]
     [-keepShape [util]]
```

Specifies the floorplan dimensions by size; or by die, I/O, or core coordinates.

When you use the Specify floorplan form (or specify the floorplan through the `floorplan` command), the floorplan is resized automatically—relatively floorplan constraints are automatically derived on the fly for blocks, fixed standard cells, fixed pre-routes, and blockages.

The floorplan is linearly adjusted as follows:

- Spacing among blocks is evenly adjusted
- The size of the modules and black boxes is evenly adjusted.
- The fixed pre-routes and I/Os are automatically adjusted.

To support bus planning, the `floorplan` command internally uses `resizeFP` in proportional mode. While `resizeFP -proportional` does not adjust bus guides based on the new core size, the `floorplan` command does not delete any bus guide during floorplanning.

Parameters

`-b` *`die_x1 die_y1 die_x2 die_y2 io_x1 io_y1 io_x2 io_y2 core_x1 core_y1 core_x2 core_y2`*

Specifies the die size and the spacing, in micrometers, of the die coordinates.

die_x1 Specifies the lower-left x coordinate of the die.

die_y1 Specifies the lower-left y coordinate of the die.

die_x2 Specifies the upper-right x coordinate of the die.

die_y2 Specifies the upper-right y coordinate of the die.

io_x1 Specifies the lower-left x coordinate of the outside edge of the I/O box.

io_y1 Specifies the lower-left y coordinate of the outside edge of the I/O box.

io_x2 Specifies the upper-right x coordinate of the outside edge of the I/O box.

io_y2 Specifies the upper-right y coordinate of the outside edge of the I/O box.

core_x1 Specifies the lower-left x coordinate of the outside edge of the core box.

core_y1 Specifies the lower-left y coordinate of the outside edge of the core box.

core_x2 Specifies the upper-right x coordinate of the outside edge of the core box.

core_y2 Specifies the upper-right y coordinate of the outside edge of the core box.

`-coreMarginsBy {io | die}`

Specifies whether the core margins are calculated using the core-to-IO boundary or the core-to-die boundary.

`coreToLeft coreToBottom coreToRight coreToTop`

Specifies the core size and the spacing, in micrometers, between the core edge, which is the margin between the outside edge of the core (head) box.

coreToLeft Specifies the margin from the outside edge of the core box to the left.

coreToBottom Specifies the margin from the outside edge of the core box to the bottom.

coreToRight Specifies the margin from the outside edge of the core box to the right.

coreToTop Specifies the margin from the outside edge of the core box to the top.

`-d` *dieH dieW coreToLeft coreToBottom coreToRight coreToTop*

Specifies the die size and the spacing, in micrometers, between the die edge, which is the margin between the outside edge of the die.

dieH Specifies the die's height value.

dieW Specifies the die's width value.

coreToLeft Specifies the margin from the outside edge of the die to the left of the I/O boundary.

coreToBottom Specifies the margin from the outside edge of the die to the bottom of the I/O boundary.

coreToRight Specifies the margin from the outside edge of the die to the right of the I/O boundary.

coreToTop Specifies the margin from the outside edge of the die to the top of the I/O boundary.

`-dieSizeByIoHeight {max | min}`

Specifies whether the maximum I/O height or the minimum I/O height should be used for die size calculation.

`[-flip {f | s | n}]`

Specifies the orientation of the bottom row in the core area.

`f` Specifies that the first row flips from the bottom up.

`n` Specifies no row flipping

`s` Specifies that the second row flips from the bottom up.

`-fplanOrigin {center | lcorner}`

Specifies whether the origin of the floorplan should be at the center or at the lower left corner.

`-keepShape [`*`util`*`]` Specifies target utilization for the rectilinear block area.

The parameter, when specified, retains the original shape of a rectilinear block during resize.

For more information on resizing the recitlinear blocks in floorplan, see Resizing Rectilinear Blocks in the “Floorplanning the Design” chapter of the *Encounter User Guide*.

`-overlapSameSiteRow`

Specifies that same-site rows can overlap with one another.

Single height rows must not overlap. Only double- or multiple-height rows can overlap, and in sections that are multiple of single height—any other overlap section size will cause the rows to be ignored.

`-r` *`aspectRatio`* Specifies the chip’s core dimensions as the ratio of the height divided by the width. If a value of 1.0 is used, a square chip is defined. A value of 2.0 will define a rectangular chip with height dimension that is twice the width dimension.

`rowDensity` [*`coreToLeft coreToBottom coreToRight coreToTop`*]

Specifies a row density value.

`coreToLeft` Specifies the margin from the outside edge of the core box to the left.

`coreToBottom` Specifies the margin from the outside edge of the core box to the bottom.

`coreToRight` Specifies the margin from the outside edge of the core box to the right.

`coreToTop` Specifies the margin from the outside edge of the core box to the top.

`-s` *`coreH coreW coreToLeft coreToBottom coreToRight coreToTop`*

Specifies the core size and the spacing, in micrometers, between the core edge, which is the margin between the outside edge of the core (head) box.

`coreH` Specifies the core box’s height value.

coreW	Specifies the core box's width value.
coreToLeft	Specifies the margin from the outside edge of the core box to the left.
coreToBottom	Specifies the margin from the outside edge of the core box to the bottom.
coreToRight	Specifies the margin from the outside edge of the core box to the right.
coreToTop	Specifies the margin from the outside edge of the core box to the top.

`-site` *siteName* — Specifies a core row site.

`-su` *aspectRatio* — Determines the core and module sizes by standard cell density.

stdCellDensity [*coreToLeft coreToBottom coreToRight coreToTop*]

Specifies a standard cell density value.

coreToLeft	Specifies the margin from the outside edge of the core box to the left.
coreToBottom	Specifies the margin from the outside edge of the core box to the bottom.
coreToRight	Specifies the margin from the outside edge of the core box to the right.
coreToTop	Specifies the margin from the outside edge of the core box to the top.

`-verticalRow` — Specifies that the rows in the floorplan are vertical, instead of horizontal.

Note: Support for vertical rows is a beta feature. Usage and support of this beta feature are subject to prior agreement with Cadence. Contact your Cadence representative if you have any questions.

Note: While importing the design, if the configuration variable `ui_isVerticalRow` is set to 1, by default the `floorPlan` command is invoked with the `-verticalRow` parameter.

For more information on vertical rows, see Using Vertical Rows in the "Floorplanning the Design" chapter of the *Encounter User Guide*.

fplanFlipOrRotateInstance

```
fplanFlipOrRotateInstance
     -flip {x | y} |
     -rotate {0 | 90 | 180 | 270}
     [-group]
```

Flips or rotates the selected instances.

Alternatively, you can use the *Floorplan - Edit Floorplan - Flip/Rotate Selected Instances* form in the Encounter GUI, to flip or rotate an instance.

Parameters

`-flip {x | y}`	Flips the selected instance(s) through x axis or y axis.
`-group`	Flips or rotates the bounding box which encloses all the selected instances.
`-rotate {0|90|180|270}`	Rotates the selected instance(s) by 90 or 180 or 270 degrees.

generateGuide

```
generateGuide
     {[-blackBlob [blobNames]]
     | [-s minModuleSize] [-noShrink]}
```

Displays the placement of all modules, block guides, and blackblob module guides. Saving the floorplan after generating a guide saves all module, block guide, and blackblob module guide data.

You can use this command after running blob placement.

Parameters

`-blackBlob [`*`blobNames`*`]`

Specifies the name of the blackblob(s) for generating the blackblob module guide. For multiple blackblobs, separate the names with a space.

Default: If no blackblob name is specified, the command generates guides for all blackblobs in the design.

`-noShrink`

Maintains the module area size. The size is calculated based on the placement data.

Default: If you do not specify this parameter, it will shrink the module size to that which is best for viewing module locations.

`-s` *`minModuleSize`*

Specifies the minimum number of instances contained in a module that is displayed in the generated floorplan.

Examples

- The following command moves all blob modules into the core area, based on existing blob placement:

  ```
  generateGuide -blackBlob
  ```

- The following command moves the blob module named `b1` into the core area, based on existing blob placement:

  ```
  generateGuide -blackBlob b1
  ```

getDrawView

`getDrawView`

Returns the design view that was set with the `setDrawView` command.

You can use this command after `setDrawView`.

Parameters

None.

getIoFlowFlag

`getIoFlowFlag`

Gets the current I/O row flow flag setting.

To set the I/O row flow flag, use the `setIoFlowFlag` command.

Parameters

None.

getNetWeight

```
getNetWeight
     { -all | netName ... }
```

Retrieves the net weight values for the specified nets.

You can use this command after running the `specifyNetWeight` command.

Parameters

`-all`	Retrieves the net weight value for all nets.
`netName`	Retrieves the net weight value for all specified nets.

getObjFPlanBoxList

```
getObjFPlanBoxList
     {Cell | Group | Instance | Layershape | Module | Net | Pin}
     ObjectName
```

Retrieves a box list of the specified rectilinear object that was created earlier with the `setObjFPlanBoxList` command. This command can also be used to retrieve a box list of a rectilinear block instance from the PEF overlap layer. A box list comprises two or more boxes.

For a graphical representation of the definition of bounding boxes, see Defining the Bounding Box in the “Floorplanning the Design” chapter of the *Encounter User Guide*.

You can use this command after setting the rectilinear shape of a block instance with the `setObjFPlanBoxList` command.

Parameters

objectName	Specifies the name of the object for object type.
{Cell | Group | Instance | Layershape | Module | Net | Pin}	Specifies the type of object for *objectName*.

Example

The following command retrieves the rectilinear boundary for module `xyz`.

```
getObjFPlanBoxList Module xyz
```

getObjFPlanPolygon

```
getObjFPlanPolygon
     ObjectType
     ObjectName
```

Retrieves polygon coordinates of the specified rectilinear object that was created earlier with the `setObjFPlanPolygon` command.

You can use this command after setting the polygon coordinates of a rectilinear block with the `setObjFPlanPolygon` command.

Parameters

objectName	Specifies the name of the object for *objectType*.
objectType	Specifies the type of object for *objectName*. This can be: ■ Cell ■ Group ■ Instance (blackbox) ■ Layer shape ■ Module

Example

The following command retrieves the polygon coordinates of module `xyz`.

```
getObjFPlanPolygon module xyz
```

getPlanDesignMode

```
getPlanDesignMode
     [-abSpacingX]
     [-abSpacingY]
     [-autoPowerPlan]
     [-boundaryPlace]
     [-congAware]
     [-fixPlacedMacros]
     [-groupHardMacro]
     [-groupIOLogic]
     [-handleFlat]
     [-keepGuide]
     [-maxDistToGuide]
     [-noColorize]
     [-numSeed]
     [-powerAware]
     [-quiet]
     [-seedSize]
     [-setSeedHierLevel]
     [-spacingX]
     [-spacingY]
     [-useExistingPowerRail]
     [-util]
```

Returns the following information about a specified Masterplan mode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software returns information for all of the Masterplan mode parameters.

Parameters

parameter_names	Returns information for the specified parameters. You can specify one or more parameters. See setPlanDesignMode for descriptions of the Masterplan mode parameters you can specify.

`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setPlanDesignMode` parameters in Tcl list format.

Examples

- The following command returns the current settings for the `-boundaryPlace` and `-congAware` parameters:

  ```
  getPlanDesignMode -boundaryPlace -congAware
  ```

  The software returns the following information:

  ```
  -boundaryPlace false                  # bool, default=false
  -congAware false                      # bool, default=false
  {boundaryPlace false} {congAware false}
  ```

- The following command returns the current setting for the `-congAware` parameter in Tcl list format:

  ```
  getPlanDesignMode -congAware -quiet
  ```

  The software returns the following information:

  ```
  false
  ```

- The following command returns the current settings for the `-boundaryPlace` and `-congAware` parameters in Tcl list format:

  ```
  getPlanDesignMode -boundaryPlace -congAware -quiet
  ```

  The software returns the following information:

  ```
  {boundaryPlace false} {congAware false}
  ```

- The following command returns the current settings for all `setPlanDesignMode` parameters in Tcl list format:

  ```
  getPlanDesignMode -quiet
  ```

  The software returns the following information:

  ```
  {abSpacingX -1} {abSpacingY -1} {autoPowerPlan 0} {boundaryPlace false}
  {congAware false} {createFence true} {fixPlacedMacros false} {groupHardMacro
  false} {groupIOLogic false} {maxDistToGuide 0.0625} {noColorize false}
  {numSeed 20} {powerAware false} {seedSize 128} {setSeedHierLevel -1} {spacingX
  -1} {spacingY -1} {useExistingPowerRail true} {util 0.75}
  ```

getUserDataAlias

`getUserDataAlias` *`variableName`*

Displays the original variable name of the specified variable.

You can use this command to retrieve the original name of a registered global variable.

Parameters

`variableName`	Specifies the name of the global variable.

getUserDataDefaultValue

`getUserDataDefaultValue` *`variableName`*

Displays the default value of the specified variable.

You can use this command after invoking Encounter.

Parameters

`variableName`	Specifies the name of the global variable.

getUserDataDesc

`getUserDataDesc` *`variableName`*

Displays the description of the specified variable.

You can use this command after invoking Encounter.

Parameters

`variableName`	Specifies the name of the global variable.

getUserDataName

`getUserDataName` *`variableNamePattern`*

Displays a list of variables that match a pattern.

You can use this command after invoking Encounter.

Parameters

`variableNamePattern`

Specifies the pattern for which to search. The pattern is specified by wildcards (?*).

Example

The following command displays the variables that start with `setPtn`.

```
getUserDataName defOut*
```

getUserDataRange

`getUserDataRange` *`variableName`*

Displays the allowed data range of the specified variable.

You can use this command after invoking Encounter.

Parameters

`variableName`	Specifies the name of the global variable.

getUserDataType

`getUserDataType` *`variableName`*

Displays the data type of the specified variable.

You can use this command after invoking Encounter.

Parameters

`variableName`	Specifies the name of the global variable.

getUserDataValue

`getUserDataValue` *`variableName`*

Displays the value of the specified variable.

You can use this command after setting the global variable.

Parameters

`variableName`	Specifies the name of the global variable.

initCoreRow

```
initCoreRow
```

Regenerates rows for the core area and all the power domains in a design with rows based on the current row parameters instead of using the specify floorplan command.

You can use this command after importing the design.

Parameters

None

initNdr

```
initNdr
     [-cloneFrom sourceRuleName]
```

Copies rules from an existing LEF default rule or a non-default rule. These rules can be edited/modified from a temporary structure.

Parameters

`-cloneFrom` *`sourceRuleName`*	Specifies the name of the non-default rule or LEF default rule that is to be cloned.

Example

The following command copies rules from an existing non-default rule `testndr`.

```
initNdr -cloneFrom testndr
```

Related Topics

- "Using the NanoRoute router" chapter in the *Encounter User Guide*
  - "Using Non-Default Rules"

legalizeFPlan

```
legalizeFPlan
     [-checkOri]
     [-checkSite]
```

Legalizes partition locations according to standard cell row orientations and specific design constraints.

You can use this command after specifying the floorplan.

Parameters

`-checkOri`	Specifies that all partitions snap to the closest `R0` row on the grid. For this rule, the Encounter software assumes that standard cell rows start with an `R0` orientation and continue in an `R0`-to-`MX` pattern.
`-checkSite`	Specifies that partitions be placed on grids so that they are legal for all design constraints.

Example

The following command legalizes all the partition locations such that they are snapped to the closest `R0` rows:

```
legalizeFPlan -checkOri
```

loadBlackBlobNetlist

`loadBlackBlobNetlist` *`verilogFileName`*

Loads a netlist for a black blob. You can use this command to incrementally load an updated netlist for a blob. The existing blob area should be equal to or larger than the area generated by the updated netlist.

if you want to retain the module as a blob object after loading a new blackblob netlist:, do the following:

- Re-specify the blob area if the existing block area is smaller than the area generated by the updated netlist.
- Perform blob placement again.

If you want to convert a blob instance back to a hierarchical module, run the `unspecifyBlackBlob` command.

Note: A netlist can be loaded only for a blackblob instance, and not for a blackblob module. If the blob is currently a blob module, run the `addHaloToBlock` command before running the `loadBlackBlobNetlist` command.

You can run this command after specifying a blackblob.

Parameters

`verilogFileName`	Specifies the name of the Verilog file that contains the updated blackblob netlist. The netlist can contain more than one blackblob netlist, but it should not contain any non-blackblob modules.

Example

The following command incrementally loads the Verilog file `results_conv_v`.

```
loadBlackBlobNetlist results_conv.v
```

loadFPlan

```
loadFPlan
     fileName
     [-noSnap]
     [-noEqualizePtnHInst]
```

Loads a floorplan file.

Note: When you run the `loadFPlan` command, the Encounter software removes any existing floorplan information and reads in the new information.

Important

Blocks and instances loaded with this command are set as preplaced.

You can use this command after importing the design.

Parameters

fileName	Specifies the name of the floorplan file that was saved.
-noEqualizePtnHInst	Disables the snapping capability of the clone partitions to the power grid, such that clones do not necessarily have the same row structure and pattern as their master.
-noSnap	Prevents snapping of a floorplan object. This option is used to load a floorplan of a partition that is rotated orthogonally to the standard cell row orientation of the top level. This partition is one of the clones of a repeated partition.

Example

The following command loads the floorplan file `myFPlan.fp`, which was saved in an earlier session:

```
loadFPlan myFPlan.fp
```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Module Constraint Types"
  - "Viewing the Floorplan"

- Load and Check Data in the *Encounter Flat Implementation Flow Guide*
- Load and Check Data in the *Encounter Hierarchical Implementation Flow Guide*

loadIoFile

```
loadIoFile
     ioFileName
     [-noAdjustDieSize]
```

Loads an I/O assignment file.

Note: By default, the loadIoFile command will automatically adjust the die size to accommodate all the IOs. If this command is used after floorplanning, it would change the die size defined in the floorplan file. If you do not want the die size to be automatically adjusted, use the `-noAdjustDieSize` parameter.

You can use this command after importing the design.

Parameters

`ioFileName`	Specifies the name of the I/O assignment file.
`-noAdjustDieSize`	Specifies that the die size should not be adjusted automatically.

Example

The following command loads the I/O assignment file `myPinFile.io`:

```
loadIoFile myPinFile.io
```

Related Topics

- "Data Preparation" chapter of the *Encounter User Guide*
  - "Generating the I/O Assignment File"

loadUserDataFile

`loadUserDataFile` *`fileName`*

Loads the global variable settings from a file previously saved through the `saveUserDataFile` command.

You can use this command after saving the global variable settings in a file through the `saveUserDataFile` command.

Parameters

`fileName`	Specifies the name of the file from which to load the variable settings.

modifyNdrViaList

```
modifyNdrViaList
     {-minCuts number [-layer layerList]} |
     {-add | -remove | -replaceAll viaList}
```

Modifies/edits the via list based on the non-default rules that you define.

Parameters

`-minCuts` *`number`*	Specifies the minimum number of cuts that will be used to prune vias from the existing via list.
`-layer` *`layerList`*	Specifies the cut layers on which the minimum cut values are applied.
`-add`	Adds a new via to the via cell list.
`-remove`	Removes a via from the via cell list.
`-replaceAll`	Replaces all the vias from the via cell list.
`viaList`	Specifies the list of vias that is to be modified.

Example 1

The following command removes via23 and via34 from the existing via list.

```
modifyNdrViaList -remove {via23 via34}
```

Example 2

The following command replaces all the existing vias with the specified via list.

```
modifyNdrViaList -replaceAll {via1 via2 via3}
```

Related Topics

- "Using the NanoRoute router" chapter in the *Encounter User Guide*
  - "Using Non-Default Rules"

moveGroupPins

```
moveGroupPins
     -loc x y
     [-layer layerId]
     [-depth d]
     [-width w]
     [-noFixed]
     [-avoidOverlap]
```

Changes the pin layer, the pin size, pin status, resolves pin overlap, and moves a selected pin or pin group to a specific location.

You can use this command after selecting the pin(s) in the move mode (selecting pins in the design display area with the *Move/Resize/Reshape* tool widget).

Parameters

`-avoidOverlap`	Specifies that pin(s) will be moved to avoid overlaps, if overlaps are present.
`-depth d`	Specifies the depth of the pin or pin group, in micrometers.
`-layer layerId`	Specifies the number of the metal layer for the pin or pin group.
`-loc x y`	Specifies the new x and y location for the pin or pin group.
`-noFixed`	Specifies that the moved pins or group of pins are not set to the *fixed* status.
`-width w`	Specifies the width of the pin or pin group, in micrometers.

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Editing Pins"

moveSelObj

`moveSelObj` *`llx lly`*

Moves a selected object in the design display area to a specific location.

You can use this command after importing the design.

Parameters

`llx`	Specifies the lower left x coordinate of the object.
`lly`	Specifies the lower left y coordinate of the object.

multiPlanDesign

```
multiPlanDesign
     [-autoTrials number]
     [-constraints {listOfConstraintFiles} {on | on_off}]
     [-congAware {on | on_off}]
     [-setSeedHierLevel minLevel maxlevel]
     [-numSeed min max step]
     [-groupIOLogic {on | on_off}]
     [-groupHardmacro {on | on_off}]
     [-boundaryPlace {on | on_off}]
     [-keepGuide {on | on_off}]
     [-maxDistToGuide min max step]
     [-keepFP dir]
     [-keepNumFP num]
     [-effort {low | medium | high}]
     [-outFile fileName]
```

Creates multiple alternative floorplans by running different variations of the `setPlanDesignMode` and `planDesign` commands. Masterplan then analyzes the resulting floorplans, and ranks their usability using estimated wire length, congestion, and timing criteria.

The `multiPlanDesign` command generates floorplans based on the parameter settings you specify. For some parameters, you can instruct Masterplan to always perform the functionality (`on`), or to perform or not perform the functionality on floorplans in a predefined order (`on_off`).

You can run `planDesign` jobs sequentially on one machine, or in parallel on multiple host machines. To run `multiPlanDesign` in parallel, you must first use the `setDistributeHost` and `setMultiCpuUsage` commands to set up the configuration for the distributed processing. For example:

```
setDistributeHost {-local | -rsh -add {machine1 machine2}}
setMultiCpuUsage -numHosts 2
multiPlanDesign -autoTrials 2
```

Floorplan rankings are automatically displayed in a separate results form after all of the `planDesign` jobs are completed. Additionally, Masterplan saves a text report of the ranking results to a user-specified file.

If you run this command on a master instance blackbox with non-R0 orientation, it automatically converts the new orientation to R0. For more information, see Handling of Blackboxes with Non-R0 Orientation in the “Partitioning the Design” chapter of the *Encounter User Guide*.

You can run this command in distributed processing mode by using the following text commands:

- setDistributeHost
- setMultiCpuUsage

Parameters

`-autoTrials` *`number`*
Specifies the number of `planDesign` jobs to run.

If you specify `-autoTrials`, you cannot specify the `-setSeedHierLevel`, `-numSeed`, and `-maxDistToGuide` parameters.

Note: If you specify the `-autoTrials` parameter, you can specify only the `on` value for boolean parameters. You cannot specify `on_off`. For example, you can specify `-boundaryPlace on`, but you cannot specify `-boundaryPlace on_off`.

Default: 1

`-boundaryPlace {on | on_off}`

Determines whether to place macros along the core boundary for chip and block designs.

If you specify `on`, Masterplan places macros along the core boundary for *all* floorplans generated.

If you specify `on_off`, Masterplan only places macros along the core boundary for some of the floorplans generated.

`-congAware {on | on_off}`

Estimates the congestion for the floorplan after macro placement.

If congestion (the Trial Route estimated Gcell overflow percentage) is greater than 0.5 percent in either the X or Y direction, Masterplan attempts to resolve it by refining the macro placement (such as changing the spacing, orientation, or grouping). Masterplan does not touch user-preplaced macros during this refinement. After re-estimating the congestion, if the congestion improved, Masterplan keeps the new floorplan. If congestion did not improve, Masterplan restores the previous floorplan.

Note: For better accuracy, it is recommended that you have power information available, to use this feature.

Default: `off`

`-constraints {`*`listOfConstraintFiles`*`} {on | on_off}`

Determines whether to use the constraints set in the specified constraint file for macro placement and module guide generation. You can specify one or more constraint files.

With a single constraint file, if you specify `on`, Masterplan uses the constraints in the file for *all* floorplans generated. If you specify `on_off`, Masterplan only uses the constraints in the file for some of the floorplans generated.

With multiple constraint files, if you specify `on`, Masterplan runs the `setPlanDesignMode` and `planDesign` commands with *`constraintFile1`* for every combination of the other specified `MultiPlanDesign` parameters. Then Masterplan repeats the process with *`constraintFile2`*, and so on.

If you specify `on_off` with multiple constraint files, Masterplan runs the following sets of commands for every combination of other specified `multiPlanDesign` parameters:

- `setPlanDesignMode` and `planDesign` without a constraint file.
- `setPlanDesignMode` and `planDesign` with *`constraintFile1`*.
- `setPlanDesignMode` and `planDesign` with *`constraintFile2`*. And so on.

Note: If you specify multiple constraint files, you can use the `on` value with `-autoTrials`, but not `on_off`.

For more information on constraint files, see "Masterplan Constraint File Format" on page 511.

`-effort {low | medium | high}`

Specifies the effort level to use for analysis. The `-effort` parameter allows you to choose between analysis accuracy and run time levels.

Default: `medium`

`low`	Provides cluster mode placement (non-timing driven). This results in low analysis accuracy, but a fast run time.

	`medium`	Provides timing driven floorplan mode placement. This results in medium analysis accuracy and run time.
	`high`	Provides timing driven default placement (`placeDesign`). This results in good analysis accuracy, but a longer run time.

`-groupHardMacro {on | on_off}`

Determines whether to control the selection of hierarchical instances as seeds, based on hard macro distribution.

If you specify `on`, Masterplan controls the selection of hierarchical instances as seeds based on hard macro distribution for *all* floorplans generated.

If you specify `on_off`, Masterplan only bases the seed selection on hard macro distribution for some of the floorplans generated.

`-groupIOLogic {on | on_off}`

Determines whether to create I/O instance group seeds by traversing the logic from the I/O pads (or pins) to the first level of sequential elements.

If you specify `on`, Masterplan creates I/O instance group seeds by traversing the logic for *all* floorplans generated.

If you specify `on_off`, Masterplan only creates I/O instance group seeds for some of the floorplans generated.

`-keepFP` *`directory`* Specifies the directory in which to store the saved floorplans.

Default: `current_working_directory/``mfp`

`-keepGuide {on | on_off}` Retains Masterplan-generated module guides after the floorplan has been created.

Default: `off`

`-keepNumFP` *`number`* Specifies the number of best floorplans to save out of the ones created.

Default: 3

`-maxDistToGuide` *`min max step`*

Restricts macro placement to an area that is relative to the macro's associated guide boundary.

If you specify this parameter, Masterplan will restrict placement to the minimum area value specified (*min*) for a floorplan. For each consecutive floorplan generated, it restricts placement to an area value that is *step* more than the previous area value, until it reaches the maximum area value specified (*max*).

Note: If you specify `-maxDistToGuide`, you cannot specify the `-autoTrials` parameter.

`-numSeed` *min max step*

Forces automatic seed selection to select approximately a specified number of seeds.

If you specify this parameter, Masterplan will select approximately the minimum number of seeds specified (*min*) for a floorplan. For each additional floorplan generated, it selects a seed number that is *step* more than the previous seed number selected. Masterplan will continue to create floorplan until it selects the maximum seed number specified (*max*).

For example, if you specify:

```
-numSeed 20 50 10
```

Masterplan selects approximately 20 seeds for one floorplan, 30 seeds for another floorplan, 40 seeds for another floorplan, and 50 seeds for the final floorplan.

Note: If you specify `-numSeed`, you cannot specify the `-autoTrials` parameter.

`-outFile` *`fileName`*

Generates a text report of the floorplan ranking results with the specified name. The text report includes information such as the wire length and congestion values, and the commands that were run for each saved floorplan.

For example:

```
################################################
###FLOORPLAN RANKED BY WIRE LENGTH ESTIMATION###
################################################
##No.1 best result
##Wire Length: 2.467297e+07
##Congestion Overflow: 0.24%H + 0.11%V
##Floorplan file: /home/pfd/mptest/mp_fp0.fp
##Commands
    setPlanDesignMode -reset
    setPlanDesignMode -congAware true -setSeedHierLevel 1
    planDesign -constraints constraintfile
##No.2 result
...
```

Default: *`topcell_name`*`.mfp`

`-setSeedHierLevel` *`minLevel maxLevel`*

Forces automatic seed selection to choose all hierarchical instances at a specific logic level as seeds.

When you specify this parameter, Masterplan will select all hierarchical instances at the specified *minLevel* logic level as seeds for the first floorplan, as long as the instances also meet the other seed criteria. For each consecutive floorplan generated, Masterplan selects all hierarchical instances at the next logic level (current logic level + one level) as seeds, until it reaches the maximum logic level specified (*maxLevel*). The top design cell has a logic level of `0`.

For example, if you specify:

```
-setSeedHierLevel 1 3
```

Masterplan selects hierarchical instances at logic level 1 as seeds for the first floorplan, hierarchical instances at logic level 2 as seeds for the second floorplan, and hierarchical instances at logic level 3 as seeds for the third floorplan.

You should not specify a *maxLevel* value that is lower than the deepest logic level in your netlist.

Note: If you specify `-setSeedHierLevel`, you cannot specify the `-autoTrials` parameter.

Related Topics

- Creating Multiple Alternative Floorplans in the “Creating An Initial Floorplan Using Masterplan” chapter of the *Encounter User Guide*.
- “Accelerating the Design Process by Using Multiple-CPU Processing” chapter of the *Encounter User Guide*

orientateInst

```
orientateInst
     inst
     {R90 | R180 | R270 | MX | MY}
```

Rotates an instance around its origin, or flips an instance mirrored through its x or y axis.

If you rotate a master instance blackbox to non-R0 orientation, the new non-R0 orientation will automatically be converted to R0 orientation. For more information, see Handling of Blackboxes with Non-R0 Orientation in the “Partitioning the Design” chapter of the *Encounter User Guide*.

You can use this command after importing the design.

Parameters

inst	Specifies the name of the hierarchical instance to be rotated or flipped.
MX	Flips the instance mirrored through its x axis.
MY	Flips the instance mirrored through its y axis.
R90	Rotates the instance around its origin 90 degrees counterclockwise.
R180	Rotates the instance around its origin 180 degrees counterclockwise.
R270	Rotates the instance around its origin 270 degrees counterclockwise.

Examples

- The following command rotates the block SH17/I441 90 degrees counterclockwise:

  ```
  orientateInst SH17/I4421 R90
  ```

- The following command flips the block SH17/I441 mirrored through its x axis:

  ```
  orientateInst SH17/I4421 MX
  ```

placeMacroInsideModule

```
placeMacroInsideModule
     hinstName
```

Moves all hard macros that belong to the specified module into its boundary.

You use this command after the module is placed in the core area.

Parameters

hinstName	Specifies the name of the hierarchical instance. You can use wildcards (*?) with this parameter.

Example

The following command moves all hard macros that belong to modules that have hierarchical instance names that match DTMF_INST/*_TEST_INST into its boundary:

```
placeMacroInsideModule DTMF_INST/*_TEST_INST
```

placePadIO

```
placePadIO
     [-rows numOfRow]
     [-maxIOHeight variable]
```

Places the I/O pads evenly from one row to multiple rows.

You use this command after loading a configuration file (`loadConfig`), when the Encounter software automatically places the I/O pads only in one row.

Parameters

`-rows` *`numOfRows`*	Specifies the number of I/O rows needed for the placement.
`-maxIOHeight`	Specifies the maximum height used to calculate the head box. By default, this command uses the minimum height.

Example

The following command places I/O pads in two I/O rows:

```
placePadIO -row 2
```

planDesign

```
planDesign
     [-constraints constraint_file]
     [-genTemplateOnly outputFile]
```

Generates a quick, initial floorplan that can be used as a starting point for making the final floorplan. Use the `planDesign` command to create multiple alternative floorplans. You can then test the floorplans to find the one that gives you the best placement and routing results.

When specified, the `planDesign` command performs the following internal functions in order:

- Selects the floorplan objects (called seeds) to be placed.
- Places the seeds.
- Refines the seeds.
- Places the macros.

You can use the `planDesign` command after importing a design, loading an initial floorplan (Encounter floorplan file or DEF file), and setting Masterplan global parameters (`setPlanDesignMode`).

If you run this command on a master instance blackbox with non-R0 orientation, it automatically converts the new orientation to R0. For more information, see Handling of Blackboxes with Non-R0 Orientation in the “Partitioning the Design” chapter of the *Encounter User Guide*.

Parameters

`-constraints` *`constraint_file`*

Uses the constraints set in the specified Masterplan constraint file for fence creation, macro placement and module guide generation.

A constraint file is a text file that can contain a list of seeds for Masterplan to use when generating fences and module guides, and basic relative, spacing and orientation constraints to follow during macro placement. For more information, see “Masterplan Constraint File Format” on page 511.

`-genTemplateOnly` *`outputFile`*

Generates a template of the Masterplan constraint file in the current working directory. When specified, the software creates the file only; it does *not* generate a floorplan (that is, it does not run `planDesign`).

If you do not specify a name, the software creates a file named constr.tmp.

Masterplan Constraint File Format

A Masterplan constraint file is a text file that can contain the following optional sections:

- Seed Section

The seed section allows you to specify the names of the hierarchical modules, hard macros, and instance groups that you want to choose as seeds. You can specify utilization values for individual modules or instance groups. The Masterplan seed refinement routine uses the specified module (seed) utilizations when generating module guides.

You can specify a seed section using the following format:

```
BEGIN SEED
seed_name [utilization_value] [Fence [minGap]]
END SEED
```

Where:

`seed_name`	Specifies the name of the hierarchical module, hard macro, or instance group that you want to choose as a seed. If you specify the `Fence` keyword, *`seed_name`* must be a hierarchical module.
`utilization_value`	Specifies the utilization value for the specified module or instance group.

`Fence [minGap]` — Indicates that a fence should be generated for the hierarchical module.

You can specify a *minGap* value to set the minimum amount of spacing allowed between fences.

Note: Masterplan checks for minimum gap violation in automatic fence creation flow for MSV designs. The `planDesign` command issues a warning message if a fence (representing power domain) or a hard macro is placed within the minimum gap range set for another power domain.

For example:

```
BEGIN SEED
Module-A/A1/abc      0.75
Module-B/B2/xyz
...
HM2
Module-C/C1/HM2
...
instGrp-1            0.85
instGrp-2
END SEED
```

- Macro Section

The macro section allows you to set spacing and orientation constraints that Masterplan uses during macro placement. You can use the orientation constraints to further restrict SYMMETRY values that are specified in the LEF file. However, the constraints you specify *cannot* conflict with the LEF SYMMETRY values.

You can specify a macro section using the following format:

```
BEGIN MACRO
cellOrientation cellName [R0] [MX] [MY] [R180] [MX90] [R90] [R270] [MY90]
instOrientation instName [R0] [MX] [MY] [R180] [MX90] [R90] [R270] [MY90]
fenceSpacing fenceName minX minY
cellSpacing cellName minX minY
instSpacing instname minX minY
END MACRO
```

Where:

`cellOrientation` *`cellName`* `[R0] [MX] [MY] [R180] [R90] [R270] [MY90]`

Specifies the orientation for all macros in the specified the cell. You must specify at least one orientation value.

`instOrientation` *`instName`* `[R0] [MX] [MY] [R180] [R90] [R270] [MY90]`

Specifies the orientation for the specified macro instance. You must specify at least one orientation value.

`fenceSpacing` *`fenceName minX minY`*

Specifies the minimum horizontal and vertical spacing for all macros in the specified fence.

`cellSpacing` *`cellName minX minY`*

Specifies the minimum horizontal and vertical spacing for all macros in the specified cell.

`instSpacing` *`instName minX minY`*

Specifies the minimum horizontal and vertical spacing for the specified macro instance.

For example:

```
BEGIN MACRO
cellOrientation RAM1 R0 R180
instOrientation C/HM2 MX MY
...
fenceSpacing bib/fps 6 6
cellSpacing rf_128x105r1w1 8 8
instSpacing fifo0/fifo/wds_fifo/fifo/fifo_high 12 12
...
END MACRO
```

- Relative Constraint Section

The relative constraint section allows you to set basic relative floorplan constraints for the placement and grouping of objects. The objects specified in the relative constraint section must first be defined as seeds in the seed section.

You can specify a constraint section using the following format:

```
BEGIN CONSTRAINT
MPPlace HinstName {T | B | L | R | TL | TR | BL | BR | (x,y)}
```

```
MPGroup GroupName {Hinst1 Hinst2 ...} [Soft | Hard]
END CONSTRAINT
```

Where:

`MPPlace HinstName {T | B | L | R | TL | TR | BL | BR | (x,y)}`

Specifies that an object should be placed as close as possible to a specified location inside the core area, or to a side or corner of the core boundary.

`HinstName` Specifies the name of a hierarchical instance or macro.

`{T | B | L | R | TL | TR | BL | BR}`

Specifies the side or corner of the core boundary. You can specify only one side or corner value per constraint.

`(x,y)` Specifies a set of coordinates within the core area.

`MPGroup GroupName {Hinst1 Hinst2 ...} [Soft | Hard | Fence]`

Specifies how objects in a group should be placed in relation to each other.

`GroupName` `{`*`Hinst1 Hinst2 ...`*`}`

Specifies the name of the group, and the names of the objects that make up the group.

You can define the following objects as part of a group: hierarchical instances, macros, and I/O pads. An object can belong to multiple groups. Macros and I/O pads can be preplaced.

`Soft` Indicates that objects within the group can be separated by non-group objects, but must still be close to each other.

`Hard` Indicates that objects within the group must be placed next to each other.

For example:

```
BEGIN CONSTRAINT
```

```
MPPlace ABC TL
MPPlace abc (4000, 6000)
MPGroup X {x1 x2}
MPGroup Y {y1 y2}
END CONSTRAINT
```

Examples

- The following example shows the format of Masterplan constraint file template that is generated with the `-genTemplateOnly` parameter:

```
##########################################
# Masterplan User Constraint File Template
##########################################
# Seed Section (optional)
# HinstName [utilization] [Fence [minGap]]
# For example:
# BEGIN SEED
# aaa
# bbb 0.85
# ccc 0.75 Fence 15
# END SEED
#
# Macro Section (optional)
# cellOrientation [R0] [MX] [MY] [R180] [MX90] [R90] [R270] [MY90]
# instOrientation [R0] [MX] [MY] [R180] [MX90] [R90] [R270] [MY90]
# fenceSpacing
#Apply to all macros in the fence
# cellSpacing
# Apply to all macros of master cell
# instSpacing
# Apply to a macro instance
# For example:
# BEGIN MACRO
# cellOrientation ABC [R0] [MX] [MY] [R180]
# instOrientation abc [R90] [MX90] [MY90] [R270]
# fenceSpacing Fence_1 6 6
# cellSpacing XYZ 8 8
# instSpacing xyz 12 12
# END MACRO
#
# Relative Constraint Section (optional)
```

```
# MPPlace HinstName T | B | L | R | TL | TR | BL | BR | (x, y)
# MPGroup GroupName {Hinst1 Hinst2 ..} [Soft | Hard]
# For example:
# BEGIN CONSTRAINT
# MPPlace ABC TL
# MPPlace abc (4000, 6000)
# MPGroup X {x1 x2}
# MPGroup Y {y1 y2} Hard
# END CONSTRAINT
```

Related Topics

- "Creating an Initial Floorplan Using Masterplan" chapter of the *Encounter User Guide*
  - "Creating an Initial Floorplan"

preplaceAllBlocks

```
preplaceAllBlocks
```

Changes the placement status of all blocks in the current design to preplaced.

You can use this command after running placement or loading a placement file.

Parameters

None

queryFPlanObject

```
queryFPlanObject
     [-pd]
```

Prints the properties of selected objects to the console. This command supports only instances and modules.

You can use this command after selecting one ore more instances or modules.

Parameters

`-pd`	Use this option to output power domain information, as well as check if the objects belong to a power domain.

Example

The following commands select and print property information for module `modA`:

```
selectInst modA
queryFPlanObject -pd
```

refineMacro

```
refineMacro
     [ -permutePack
      | -area llx lly urx ury [-selected]
      | -markStep
      | -restoreStep step_number
      | -restoreMark mark_number]
     [-adjustPack]
     [-ioPinClearance]
```

Refines the placement of macros in the floorplan. By default, the `refineMacro` command performs global incremental macro adjustment. You can also adjust specific macro packs, or all of the macros in a specific area of the design.

You can use `refineMacro` command after creating an initial floorplan and analyzing the results. Macros must have a status of `PLACED` in the database in order to use this command.

Parameters

`-adjustPack`
: Adjusts the way that macros packs are placed, relative to their locations. For example, an array of 16 macros might be placed in four rows of four macros (4 x 4), if they are near the top or bottom of the chip. This might change to eight rows of two macros (8 x 2), if the macros are moved near the left or right edge of the chip.

`-area` *`llx lly urx ury`*
: Adjusts the placement of any macros within the specified area to reduce empty area between them.

  If you also specify `-selected`, Masterplan moves the selected macros to the specified area and adjusts their position along with any macros already within the area.

  Type: Float, specified in microns

`-ioPinClearance`
: Leaves an internally calculated amount of clearance space between placed macros and block I/O pins. Use this parameter to ensure that pins are accessible to the router.

`-markStep`

Marks an intermediate macro refinement step to save.

When you perform incremental macro refinement, each adjustment is considered a step. By default, Masterplan saves all incremental steps done with the `refineMacro` command and lists them in the log file with a number. It does not save steps done through manual refinement (such as move, rotate, and flip).

You can use this parameter to mark manual refinement steps, or any other important step that you want to save.

Note: Marked steps are also listed in the log file by number. Masterplan saves up to1000 incremental refinement steps in the database and 20 marks before overwriting them.

`-permutePack`

Adjusts the placement of a selected pack of macros to reduce empty area between them. A pack is a set of macros from the same seed that have similar sizes and aspect ratios and have been grouped together.

You must select the macro pack in the main window before using this parameter. You can select one macro to select the entire pack.

Important

Macro pack information is stored temporarily in the data base. If you quit the Encounter session, the information is lost.

`-restoreMark` *`mark_number`*

Returns the design to the state it was in after the specified marked step (`-markStep`).

For example, to return the design to the state it was in after the step saved as `mark2`, specify:

```
refineMacro -restoreMark mark2
```

You can find the mark numbers listed in the log file.

`-restoreStep` *`step_number`*

Returns the design to the state it was in after the specified intermediate non-marked step.

For example, to return the design to the state it was in after step 4, specify:

```
refineMacro -restoreStep step4
```

To return the design to the state it was in after the previous adjustment, specify:

```
refineMacro -restoreStep -1
```

You can find the step numbers listed in the log file.

`-selected`

Moves the selected macros to the specified area and adjusts their position along with any macros already within the area.

Note: You cannot use this parameter without specifying the `-area` parameter.

relativeFPlan

```
relativeFPlan
     [--place
          objName
          quadrant
          refx refy
          xspace yspace
          [orient]
     [--relativePlace
          objName
          quadrant
          ref[h]InstName
          instName
          refcorner
          xspace yspace
          [orient]
     [--reshape
          objName
          fixedcorner
          constraineddim
          constraint
          offset
          [constraint2 offset2]
     [--relativeReshape
          objName
          fixedcorner
          constraineddim
          ref[h]instName
          [offset]
     [--reshapeBetween
          objName
          fixedcorner
          constraineddim
          ref[h]instName1 ref[h]instName2
          space
     [--reshapeCover
          objName
          fixedcorner
          constraineddim
          ref[h]instName1 ref[h]instName2
     [--preRoute
          netName
          hLayer
          vLayer
          hWidth
          vWidth
          {refObj refObjCorner xOffset yOffset | x y} ...
          shape
          state
          xRefObj
```

```
     xRefObjCorner
     yRefObj
     yRefObjCorner
[--resize
     obj
     fixedCorner
     constraineddim
     constraint
     utilization
[--relativeResize
     obj
     fixedCorner
     constraineddim
     utilization
     refObj
     offset
[--resizeBetween
     obj
     fixedCorner
     constraineddim
     utilization
     refObj1
     refObj2
     space
[--resizeCover
     obj
     fixedCorner
     constraineddim
     utilization
     refObj1
     refObj2
[--hierarchyPrefixOn
     hierarchyPrefix
[--hierarchyPrefixOff
[--updateMacros
     instName
     quadrant
     hInstName
     refcorner
     xspace yspace]
```

The `relativeFPlan` command captures and defines the placement relationship of floorplan objects independently from the actual coordinates in a floorplan, and resizes modules or blackboxes based on other floorplan objects, even outside the core boundary.

You can use this command after importing the design.

Sub Commands

This command is comprised of a set of the following sub commands, which must be preceded by two dashes (--).

`--hierarchyPrefixOff`	Disables the hierarchy prefix. After this command, all relative floorplan objects defined hereafter will use their object names without any hierarchy prefix.
`--hierarchyPrefixOn`	Applies hierarchy prefix to some hierarchy blocks which have their own relative relationships defined at block level. After this sub command, all relative floorplan objects defined hereafter will have hierarchy prefix applied.
`--place`	Used for a simple placement, this specifies that the instance name, orientation, and X/Y coordinates are passed, in addition to the quadrant placement of the object relative to the X/Y coordinates.
`--preRoute`	Specifies a wire's start or end point relative to the reference object's reference corner, or specify a wire's start or end point directly.
`--relativePlace`	Places a specific object relative to a reference object. For placement, this specifies that the X/Y coordinates and quadrant are passed, in addition to the reference corner of the reference object where the quadrant center will be.
`--relativeReshape`	Reshapes an object to be the same as the reference object in the specified dimension, but offset by the value specified in `-offset`, which can be a positive or negative number.
`--relativeResize`	Resizes an object to be the same as the reference object in the specified dimension, but offset by the value specified in `-offset`, which can be a positive or negative number.
`--reshape`	Specifies a new dimension value that a specific object will be stretched in the specified dimension from a given fixed corner.
`--reshapeCover`	Reshapes an object so that its new dimension value in the constrained dimension will be the same as the sum of the first object dimension, second object dimension, and the spacing between these two reference objects.

`--reshapeBetween`	Reshapes a given object such that its new dimension value in the constrained dimension will be equal to the spacing from the fixed reference corner to the reference object. This command does not re-position the reshaped object unless the object boundary after reshaping is extended outside of the core boundary.
`--resize`	Specifies a new dimension value that a specific object will be resized in the specified dimension from a given fixed corner.
`--resizeCover`	Resizes an object so that its new dimension value in the constrained dimension will be the same as the sum of the first object dimension, second object dimension, and the spacing between these two reference objects.
`--resizeBetween`	Resizes a given object such that its new dimension value in the constrained dimension will be equal to the spacing from the fixed reference corner to the reference object. This command does not reposition the resized object unless the object boundary after resizing is extended outside of the core boundary.
`--updateMacros`	Moves all the macros that belong to the specified module together such that their relative position will be maintained.

Parameters

Some of the following parameters apply to several subcommands, while others are for a specific subcommand. Check the syntax for the parameter correlation.

`constraint`	Specifies the new dimension value.
`constraineddim`	Specifies the constrained dimension. Possible values are `H`, `Height`, `W`, and `Width`.
`fixedcorner`	Specifies the corner of the reshaped object will be fixed during reshape. The values are `BL` for bottom left, `BR` for bottom right, `TR` for top right, and `TL` for top left.
`hLayer`	Specifies the horizontal layer of the wire segment or strip box.
`hWidth`	Specifies the horizontal wire width.
`netName`	Specifies the name of the net.

offset (for `--reshape`) Specifies the offset from the specified `constraint` value. The offset value can be a positive or negative number.

offset (for `--relativeReshape` and `--relativeResize`)

Specifies the offset value from the constrained dimension of the reference object. The offset value can be a positive or negative number.

objName Specifies the name of the object.

orient Specifies the orientation of the placed object. The values are `R0`, `MX`, `MY`, `R180`, `MX90`, `R90`, `R270`, `MY90`.

Default: If you do not specify this parameter, the current orientation is maintained.

For a key to the orientation values, see the orientation key in the "Floorplanning the Design" chapter of the *Encounter User Guide*.

quadrant Specifies the quadrant where the object will be placed. The values are `BL` for bottom left, `BR` for bottom right, `TR` for top right, and `TL` for top left.

refcorner Specifies the reference corner where the quadrant center will be placed. The values are `BL` for bottom left, `BR` for bottom right, `TR` for top right, and `TL` for top left.

ref[h]instName Specifies the name of the instance or hierarchical instance of the reference object.

{*refObj refObjCorner xOffset yOffset* | *x y*} ...

refObj specifies the reference object, which can be the core, design, or other floorplan objects.

refObjCorner specifies the corner of the reference object (*refObj*). The values are BL for bottom left, BR for bottom right, TR for top right, and TL for top left.

xOffset specifies the x offset value from *refObjCorner*. this value could be positive or negative number.

yOffset specifies the y offset value from *refObjCorner*. this value could be positive or negative number.

x y specifies the x and y coordinates for the start and end point of the wire to be created.

Note: *xRefObj*, *xRefObjCorner*, *yRefObj*, and *yRefObjCorner* parameters are rarely used. A wire could consist of many points. Each point has the location (x,y). Normally, one point has only one *refObj* and *refObjCorner*. The *xRefObj* and *yRefObj* parameters are used only when a wire point x and y, each reference to a different object. However, -shiftBased resizeFP is preferred over relativeFPlan for preserving pre-routed wires.

refx refy — Specifies the reference coordinates of the quadrant center point.

space (for --reshapeBetween and --resizeBetween)

Specifies the channel spacing between the reshaped or resized object and the reference object. This value can be a positive or negative number. If negative, the reshaped or resized objects might overlap the reference object.

vLayer — Specifies vertical layer of the wire segment or strip box.

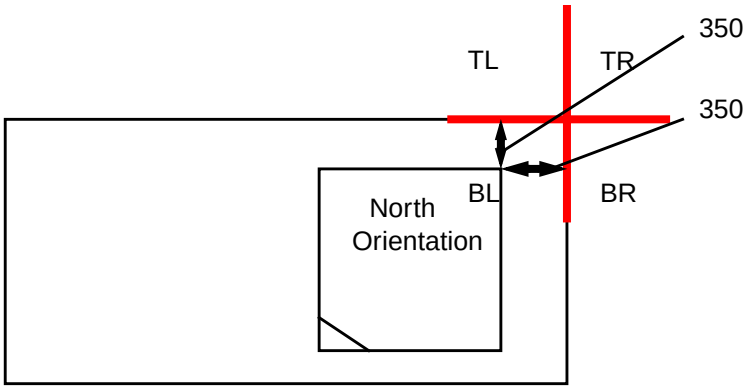
vWidth — Specifies the vertical wire width.

xspace yspace — Specifies the location where the object will be placed relative to the reference corner.

Examples

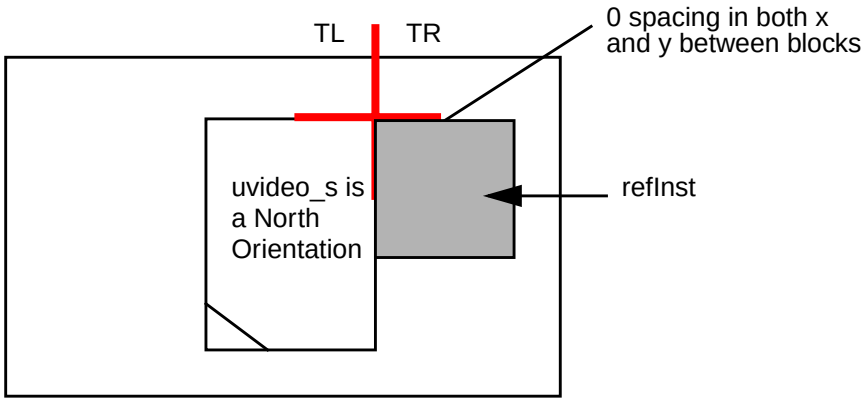
- The following command places `inst1` in the bottom-left (`BL`) quadrant, and 350 µm in both X and Y directions from the quadrant center point. The quadrant center point is at the upper-right corner of the core boundary (`1250 2020`):

  ```
  relativeFPlan –-place inst1 BL 1250 2020 350 350
  ```



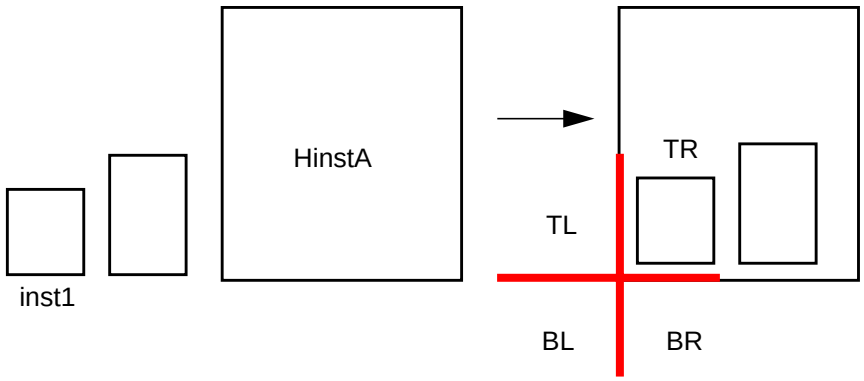
- The following command places `uvideo_s` in the bottom-left (`BL`) quadrant, and 0 µm in both X and Y directions from the quadrant center point. The quadrant center point is at the upper-left corner of `refInst`:

  ```
  relativeFPlan –-relativePlace uvideo_s BL refInst TL 0 0
  ```



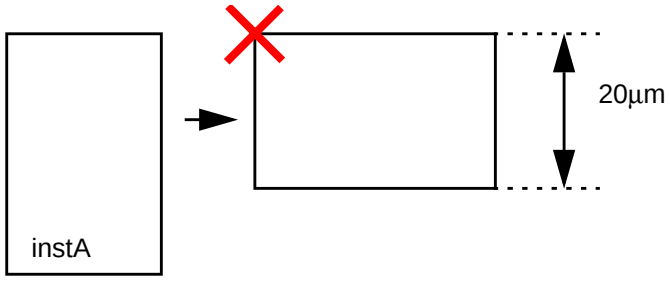
- The following command places `inst1` in the top-right (TR) quadrant, and 5 µm in both X and Y directions from the quadrant center point. The quadrant center point is at the bottom-left corner of `HinstA`:

```
relativeFPlan –-updateMacro inst1 TR HinstA BL 5 5
```



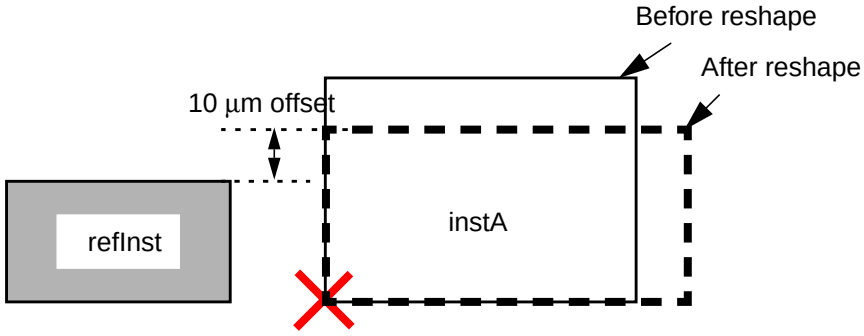
- The following command reshapes `instA` height to 20 µm with upper-left corner fixed:

```
relativeFPlan –-reshape instA TL Height 20 0
```



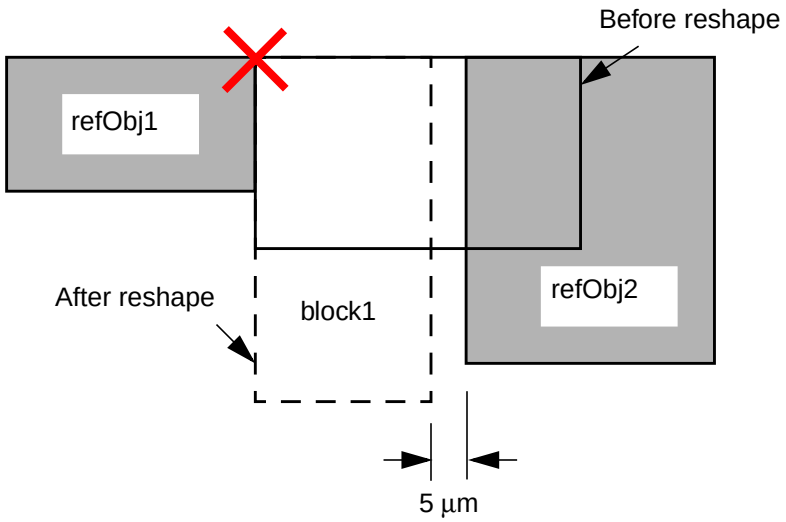
- The following command constrains the height of `instA` by the height of `refInst` plus 10 µm offset. `instA` is reshaped with lower-left corner fixed:

```
relativeFPlan  –-relativeReshape instA BL Height refInst 10
```



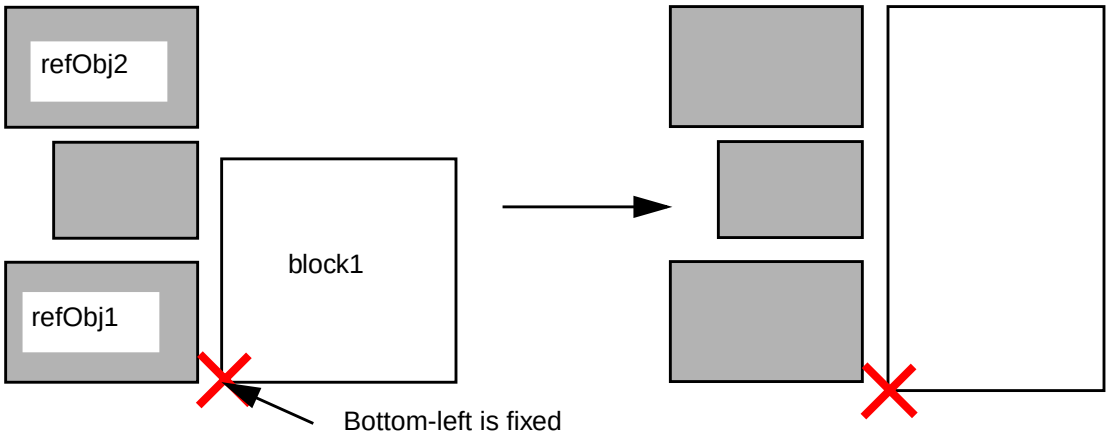
- The following command reshapes `block1` with the width dimension value that leaves 5 µm channel width between `block1` and `refObj2`. `block1` is reshaped with upper left corner fixed:

```
relativeFPlan  --reshapeBetween block1 TL Width refObj1 refObj2 5
```



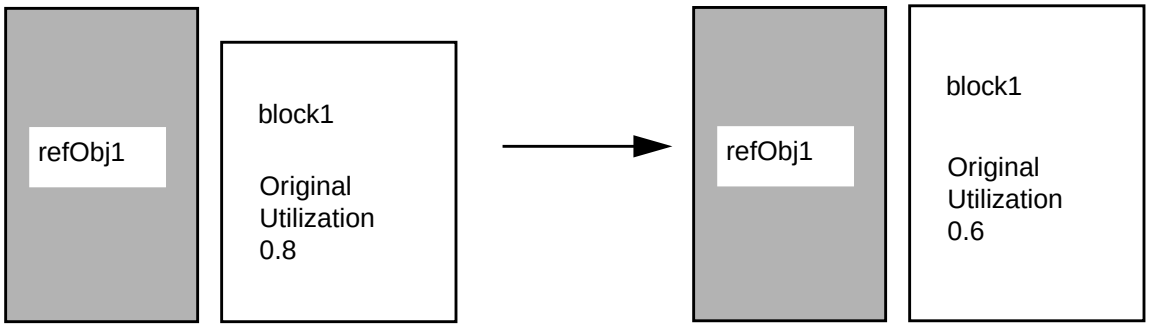
- The following command reshapes `block1` so that its new height is the same as the sum of `refObj1` and `refObj2` height, and the spacing between the two objects. `block1` is reshaped with the bottom-left corner fixed.

```
relativeFPlan  --reshapeCover block1 BL Height refObj1 refObj2
```



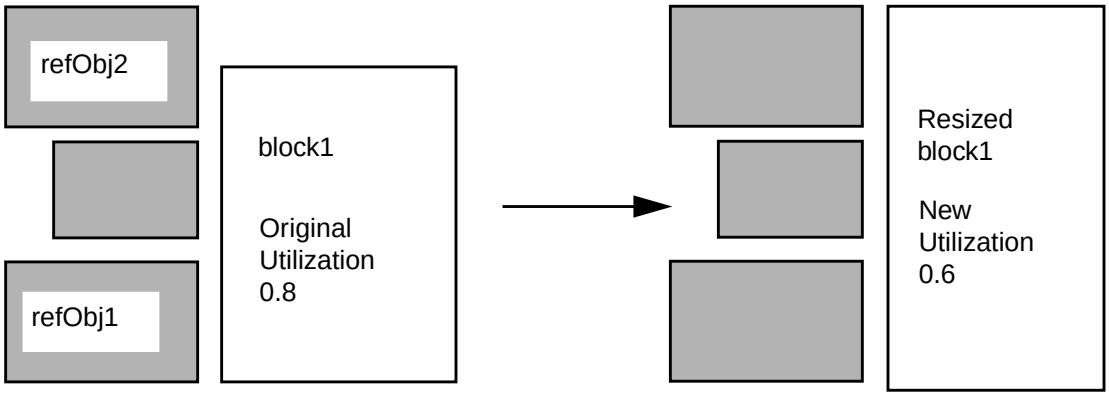
- The following command resizes `block1` so that its new height is the same as the height of `refObj1`, and moves with a block area utilization of 0.6.

```
relativeFPlan –-relativeResize block1 BL Height 0.6 refObj 0
```



- The following command resizes `block1` so that its new height is the same as the distance between `refObj1` and `refObj2`, and moves with a block area utilization of 0.6.

```
relativeFPlan –-resizeCover block1 BL Height 0.6 refObj1 refObj2
```



Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Running Relative Floorplanning"

relativePlace

```
relativePlace objName {refObjName | CORE | DESIGN}
     [-relation {L|R|T|B}]
     [-alignedBy {L|R|T|B}] [-nested]
     [-xOffset value] [-yOffset value]
     [-orientation {R0|R90|R180|R270|MX|M|MX90|MY90}]
     [-x value] [-y value]
```

Places the floorplan object relative to the reference object, or the core area or the design area.

You can use this command after importing the design.

Parameters

`-alignedBy {L|R|T|B}`

Specifies whether the object is aligned to the left, right, top, or bottom of the design area.

`CORE` — Specifies that the floorplan object will be placed relative to the core area.

`DESIGN` — Specifies that the floorplan object will be placed relative to the design area.

`-nested` — Places the specified object inside the reference object.

`objName` — Specifies the name of the floorplan object

`refObjName` — Specifies the reference object name relative to which the floorplan object will be placed.

`-relation {L|R|T|B}` — Specifies the position of the floorplan object (*`objName`*) relative to the reference object (*`refObjName`*). The floorplan object can be positioned to the left, right, top, or bottom of the reference object.

`-orientation {R0|R90|R180|R270|MX|M|MX90|MY90}`

Specifies the orientation of the floorplan object.

`-xOffset` *`value`* — Specifies the x offset value from the referenced object.

`-yOffset` *`value`* — Specifies the y offset value from the referenced object.

`-x` *`value`* — Specifies the x coordinate of the object's location.

`-y` *`value`* Specifies the y coordinate of the object's location.

Example

- The following command places the floorplan object `DTMF_INST/RAM_128x16_TEST_INST/RAM_128x16_INST` aligned to the left of the reference object `DTMF_INST/RAM_256x16_TEST_INST/RAM_256x16_INST` at (0,0):

```
relativePlace DTMF_INST/RAM_128x16_TEST_INST/RAM_128x16_INST DTMF_INST/
RAM_256x16_TEST_INST/RAM_256x16_INST -relation T -alignedBy L -xSpace 0 -
ySpace 0
```

reportNetGroup

```
reportNetGroup
     [-outfile fileName]
```

Creates a report that contains net groups and their nets.

You can use this command after importing the design.

Parameters

`-outfile` *`fileName`*	Specifies the report filename. If no filename is specified, the report is printed to the console window.

Example

Check the bump assignment and send the report to `netGrp.rpt`:

```
reportNetGroups -outfile netGrp.rpt
```

The output format is the same as the NetGroup section of the floorplan file:

```
# NetGroup Report File
NetGroup: netGroupName_one
    GroupNet net_name_01_in_netGroupName_one
    GroupNet net_name_02_in_netGroupName_one
NetGroup: netGroupName_two
    GroupNet net_name_01_in_netGroupName_two
    GroupNet net_name_02_in_netGroupName_two
```

reportPlanDesign

```
reportPlanDesign -outFile fileName.rpt
```

Note: The `reportFloorplan` command is obsolete and has been replaced by `analyzeFloorplan`. The obsolete command still works in this release, but to avoid warnings, and to ensure compatibility with future releases, update your scripts to use `analyzeFloorplan`.

Generates reports for each floorplan you create with Masterplan. You can then compare the reports to gauge which Masterplan-generated floorplan will give you the best placement and routing results.

The report includes statistics on the following items:

- Out-of-core macros and module guides
- Crossing nets between module guides, and guides to I/Os
- Wire crossings over macros and module guides
- Wire length between module guides, and guides to I/Os
- Macro placement empty area
- Module guide overlap area
- Macro-macro overlap area, and macro-blockage overlap area

Note: Items in the report have an absolute value and a percentage value, whenever possible. Because each floorplan you generate is different, not every statistic in the report will have a value for a specific floorplan.

Parameters

`-outFile fileName.rpt`

Outputs the floorplan report results to the specified file.

Examples

- The following example shows a sample floorplan report:

```
Total number of module guides is 72 (39 + 1 + 32)
Total number of hard macros is 94
Total number of ios is 172
# of nets is 234034
```

```
# of nets with connections over 40 is 415

************************ MODULE SCORES ******************************
0       : Total number of macros exceeding the core area.
0       : Total number of module guides exceeding the core area.
15994201   ( 10.2632% ) : Crossing nets among module guides.
3360   ( 5.8577% ) : Crossing nets (Io & Io).
596752   ( 13.8437% ) : Crossing nets (Io & module).
38660   ( 2.9895% ) : Guide crossings over guides.
1105   ( 6.3947% ) : Io crossings over guides.
5655   ( 0.3349% ) : Guide crossings over macros.
1043   ( 4.6232% ) : Io crossings over macros is
4.5219e+10   ( 98.3649% ) : Total guide-guide wire length.
7.5168e+08   ( 1.6351% ) : Total guide-Io wire length.
0.0   ( 0.0000% ) : Total number of dead Area.
3.8054e+10   ( 0.1401% ) : Module guide overlap area.
0.0   ( 0.0000% ) : Hard macro overlap area.
736769655440.0   ( 2.7130% ) : Hard macro overlap blockage area.
```

reportSelect

```
reportSelect
     -file filename
```

Reports the properties of one or more selected objects in the design display area in the main console window and the `encounter.log` file.

You can use this command after selecting one or more objects in the design display area.

Parameters

`-file` *`filename`*	Specifies the name of the output file that reports the properties of the selected object.

reportUnsnapBlocks

`reportUnsnapBlocks`

Reports unsnapped blocks based on the snap preferences (as defined in the *Floorplan* Tab of the Preferences form).

You can use this command after importing the design.

Parameters

None

resizeFP

```
resizeFP {[-xSize xAxisSize] [-ySize yAxisSize] [-forceResize] {-shiftBased |
     -proportional}} | -undo
     [-ioProportional]
     [-snapToTrack]
```

Resizes the floorplan while maintaining the relative locations of the existing floorplan.

- `resizeFP` resizes the space among floorplan objects proportionately or shift the floorplan objects at appropriate location(s) without changing the proportionate spacing (shift-based resize). The resize values can be snapped to the multiple integer of the metal layer pitch.
- In the shift-based mode, `resizeFP` automatically adjusts bus guides to retain it's topology. Based on the resize specifications (`-xSize, -ySize`), the bus guide shrinks or expands.

  In the proportional mode, `resizeFP` does not adjust a bus guide based on the core size. However, `resizeFP` proportional mode does not delete any bus guide either.
- `resizeFP` can also adjust the space between the I/Os proportionally, if you specify the `ioProportional` parameter. The overlapping I/O pads can be moved together as single entities when resizing the floorplan.

  Note: If an offset exists between the I/Os and the design boundary, the command preserves the offset when resizing the floorplan.

You can use this command after importing the design.

Parameters

`-forceResize`
: Resizes the floorplan even if the target size can not be met. However, the floorplan is resized as near as possible to the specified size.

  `Default:` If you do not specify this parameter, the floorplan is not resized if the target size can not be met.

  Note: This parameter can be used only with the `-shiftBased` parameter.

`-ioProportional`	Resizes the space among I/Os proportionally. *`Default`*: If you do not specify this parameter, the I/Os are distributed evenly in both, Proportional and Shift Based modes. **Note:** The overlapping I/Os are also spaced proportionally when this parameter is specified. By default, the overlapping I/Os are spaced evenly. However, the relative placement order of the I/Os and the side constraints remain unchanged.
`-proportional`	Resizes the space among floorplan objects proportionally.
`-shiftBased`	Shifts floorplan objects at appropriate location(s) without changing the proportional spacing. You can specify resize lines to control the areas of floorplan that will be expanded or shrunk. To set resize lines, use the `setResizeLine` command. If you do not specify resize line(s), the floorplan is resized automatically—that is, the resize lines are derived automatically based on the current floorplan and the specified resize values.
`-undo`	Revert to the floorplan that existed before the command was run.
`-xSize` *`xAxisSize`*	Specifies how much the floorplan should be expanded or shrunk in the x direction. A positive value expands the floorplan and a negative value shrinks the floorplan. This value is in microns. **Note:** If both `-xSize` and `-ySize` parameters are specified, the resize order is the same as the order of the parameters.
`-ySize` *`yAxisSize`*	Specifies how much the floorplan should be expanded or shrunk in the y direction. A positive value expands the floorplan and a negative value shrinks the floorplan. This value is in microns.

`-snapToTrack`

Snaps resize values (shrink/expand) of the floorplan to a multiple integer of the metal layer pitch.

For example, if the horizontal metal pitch is 1.5 microns and you want to shrink the floorplan by 8 microns in y direction, the actual shrink value is 7.84 microns, the nearest multiple integer of the metal pitch.

For HHVV layer routing technology (M1M2M3M4), this option snaps the y direction resize value to the second horizontal layer pitch (M2) and the x direction resize value to the first vertical layer pitch (M3).

For VVHH layer routing technology (M1M2M3M4), this option snaps the y direction resize value to the first horizontal layer pitch (M3) and the x direction resize value to the second vertical layer pitch (M2).

For HVHV and VHVH layer routing technology, this option snaps the y direction resize value to the first horizontal layer pitch and the x direction resize value to the first vertical layer pitch.

`Default`: Off

Example

The following command resizes the floorplan as follows: shrinks the floorplan proportionately by -49.5 microns in the x direction and -39.76 microns in the y direction, distributes the I/Os proportionally, and snaps the x direction resize value to the first vertical layer (M2) and y direction resize value to the first horizontal layer (M1).

```
resizeFP -xSize -50 -ySize -40 -proportional -ioProportional -snapToTrack
```

Related Topics

- "Floorplan Menu" chapter of the *Encounter Menu Reference*
  - Resize Floorplan
- "Floorplanning the Design" chapter of the Encounter User Guide
  - Resizing the Floorplan

restoreRelativeFPlan

`restoreRelativeFPlan`

Restores the relative floorplan information to the design display area.

You can use this command after running `relativeFPlan`.

Parameters

None

rotateInst

```
rotateInst
     instName
     {R90 | R180 | R270}
```

Rotates an instance around its origin.

If you rotate a master instance blackbox to non-R0 orientation, the new non-R0 orientation will automatically be converted to R0 orientation. For more information, see Handling of Blackboxes with Non-R0 Orientation in the “Partitioning the Design” chapter of the *Encounter User Guide*.

You can use this command after importing the design.

Parameters

instName	Specifies the name of the hierarchical instance to be rotated.
R90	Rotates the instance around its origin 90 degrees counterclockwise.
R180	Rotates the instance around its origin 180 degrees counterclockwise.
R270	Rotates the instance around its origin 270 degrees counterclockwise.

Example

The following command rotates the block SH17/I441 90 degrees counterclockwise:

```
rotateInst SH17/I4421 R90
```

runRcNetlistRestruct

```
runRcNetlistRestruct
     [-addConstantPorts]
     [-keepTempFiles]
     [-nameDelimiter delimiter]
     [-noMultiPortFix]
     [-reportFile fileName]
     [-saveToDesignName name]
```

Generates a regrouped netlist using RTL Compiler.

Note: To add logical hierarchy without creating additional hierarchy or to manipulate the logical hierarchy, see *Adding Logical Hierarchy Without Creating Additional Hierarchy* or *Logical Hierarchy Manipulation* respectively in the *Encounter User Guide*.

You can use this command after creating a new group for the physical hierarchy.

Parameters

`-addConstantPorts`

Adds constant ports and avoids additional assign nets for restructured instances. The software creates a constant port to propagate the tie high and tie low ports through the new hierarchy. A new net name is assigned to the port with an assignment of 1.

Default: The software does not propagate tie high and tie low ports.

`-keepTempFiles`

Retains the temporary files after running the `runRCNetlistRestruct` command. The software retains the following temporary files in the `./restr_input` directory:

- `restr.v`
- `restr.sdc`
- `restr.tcl`
  (script used by RTL Compiler for restructuring)

`-nameDelimiter` *`delimiter`*

Generates the hierarchical name for moved instances after restructuring. For example, if you have an original hierarchy of `/a/b/c` and want to place module `c` into a new module called `r`, using the following delimiters will enable you to retain the original hierarchy name after restructuring:

delimiter: _ (underscore)
original hierarchy: `/a/b/c`
new hierarchy: `/r/a_b_c`

delimiter: `/` (slash)
original hierarchy: `/a/b/c`
new hierarchy: `/r/a\/b\/c`

delimiter: `""` (no name delimiter)
original hierarchy: `/a/b/c`
new hierarchy: `/r/c`

Default: _ (underscore).

`-noMultiPortFix`

Specifies that no nets will be split.

Default: Splits any net that is connected to more than one output port, where each net is driven by a buffer and is connected to only one port. The software transforms the nets connected to multiple ports.

`-reportFile` *`fileName`*

Generates a report file that provides the mapping between the old and new logical hierarchy. This parameter enables you to specify the report file directory and file name.

Default: `./restr_output/Old_To_New_Hier_Map`

`-saveToDesignName` *`name`*

Specifies the new directory prefix where the data will be stored. This parameter enables you to specify a new directory and Encounter database name.

Default: `./restr_output/`*`topName`*`_restr.enc`

saveFPlan

```
saveFPlan
     fileName
     [-saveStdRow]
     [-noName]
     [-V3]
     [-V4]
```

Saves the floorplan information to a file.

Note: The command saves all the floorplan blocks that are placed/fixed/covered, in the floorplan file.

By default, the floorplan information is saved as a Version 8 floorplan file. To save a design for use with the 6.1 release, specify the -V4 parameter.

You can use this command after importing the design.

Parameters

fileName	Specifies the name of floorplan file to be saved.
-noName	Specifies that floorplan object names are not output to the floorplan file when you save the floorplan.
-saveStdRow	Saves the standard cell row definitions and the row flip status for all rows in the core design area.
-V3	Saves a Version 3 floorplan file, instead of the default release version.
-V4	Saves a Version 4 floorplan file.

Example

The following command writes the current floorplan information to the file myFloorPlan.fp:

```
saveFPlan myFloorPlan.fp
```

saveIoFile

```
saveIoFile [-v2]
     [-locations | -byOrder]
     [-temp [-ioOrder {default | clockwise | counterclockwise}]]
     [-relativeOrient]
     fileName
```

Saves the current I/O information to a Version: 3 file by default, and provides you the option to save the I/O information in a Version: 2 file for Encounter release version 6.2 and below.

This file can be used to specify the I/O Assignment File in the Design Import form. This is used to save the I/O pin or pad order to a file after assigning pins or pads by loading a floorplan file or DEF data. This I/O Assignment file is rule-based, and the keyword in the I/O file is Version: 3.

You can use this command after floorplanning the design.

Parameters

`-byOrder` Specifies the order of the I/O pad placement locations in the I/O file.

`-ioOrder {default | clockwise | counterclockwise}]`

Specifies the order of placing the I/O pins in the I/O template file. You can choose one of the following:

- `default` — Specifies the default pin order.

  For a vertical edge, the default pin order is from the bottom to the top.

  For a horizontal edge, the default pin order is from the left of the right.

- `clockwise` — Specifies the pin order in the clockwise direction.

- `counterclockwise` — Specifies the pin order in the counter-clockwise direction.

Note: For I/O cells, the `saveIoFile` command will write out only the default I/O order.

`fileName` Specifies the name of the pin order file to be written to.

`-locations`	Generates an I/O file that has placement locations rather than a sequence or order of the I/O pads. Placement co-ordinates are relative to the design boundary and are not absolute locations. *Default:* If you do not specify this parameter, it generates an order file.
`-relativeOrient`	Specifies an I/O orientation relative to the bottom (south) side.
`-temp`	Performs initial I/O placement by randomly distributing the I/Os evenly. Writes out an I/O template file with evenly spaced I/O pins, after design import. Once the I/O template file is generated, you can modify the I/O file manually and reload the file in Encounter.
`-v2`	Saves the current I/O information to a Version: 2 file for Encounter release version 6.2 and below. If you specify this parameter for saving an I/O file consisting of rectilinear design for I/O pads, Encounter displays an error message. This is because the rectilinear design for I/O pads is supported only in the current release version of Encounter. *Default*: If you do not specify this parameter, the I/O information is saved in a Version: 3 file.

Example

The following command writes the current I/O pin information to Version: 3 file:

```
saveIoFile myIoPlan.io
```

The following command writes the current I/O pin information to a Version: 2 file and specifies the clockwise placement of the I/O pins in the I/O template file:

```
saveIoFile -v2 -ioOrder clockwise -template myIoPlan.io
```

Related Topics

- "Data Preparation" chapter of the *Encounter User Guide*
  - "Generating the I/O Assignment File"

saveRelativeFPlan

```
saveRelativeFPlan fileName
```

Saves the relative floorplan constraints of floorplan objects in the current Encounter session in a file.

You can use this command after generating or editing relative floorplan constraints.

Parameters

fileName	Specifies the name of the file in which the relative floorplan constraints should be saved.

Example

The following command saves the relative floorplan constraints in the file `rfpfile10.tcl`

```
saveRelativeFPlan rfpfile10.tcl
```

saveUserDataFile

`saveUserDataFile` *`fileName`*

Saves the current global variable settings in a file.

You can use this command after setting the global variable.

Parameters

`fileName`	Specifies the file in which the global variable setting should be saved.

selectGroup

`selectGroup` *`groupName`*

Selects a group.

You can use this command after importing the design.

Parameters

`groupName`	Specifies the name of the group.

selectInst

`selectInst` *`instName`*

Selects an instance and highlights it in the design display window. You cannot specify more than one instance; however, you can use a wildcard (*).

You can use this command after importing or restoring the design.

Parameters

`instName`	Specifies the name of the instance.

selectInstByCellName

`selectInstByCellName` *`cellName`*

Selects an instance by cell name.

You can use this command after importing the design.

Parameters

`cellName`	Specifies the name of the cell.

selectInstOnNet

`selectInstOnNet` *`netName`*

Selects an instance on a net.

You can use this command after importing the design.

Parameters

`netName`	Specifies the name of the net.

selectIOPin

`selectIOPin` *`pinName`*

Selects an I/O pin.

You can use this command after importing the design.

Parameters

`pinName`	Specifies the name of the I/O pin.

selectNet

```
selectNet
     {netName | -allDefClock | -clock | -nonDefaultRule] | -shield}
```

Selects a net and highlights it in the design display window.

You can use this command after importing or restoring the design.

Parameters

`-allDefClock`	Selects all nets that have a DEF attribute `+ USE CLOCK` **Note:** The `-allDefClock` parameter is for backward compatibility only. Cadence recommends that you use the `-clock` parameter to select all nets that have a DEF attribute `+ USE CLOCK`
`-clock`	Selects all nets that have a DEF attribute `+ USE CLOCK`
`netName`	Selects the name of the net. You cannot specify more than one net. You can, however, use a wildcard (*)
`-nonDefaultRule`	Selects all nets that have a DEF attribute `+ USE NONDEFAULTRULE`
`-shield`	Selects all nets that have a DEF attribute `+ USE SHIELDNETS`

selectRouteBlk

```
selectRouteBlk
     [llx lly urx ury]
     name
     {layer1 layer2 ...}
```

Selects the routing blockages based on the following specifications:

- The coordinates of the bounding box
- The name of the blockage
- The layer(s) or the via(s). The command supports selection on multiple layers and/or vias.

If you don't specify the box coordinates, [*llx lly urx ury*], the command selects the first routing blockage by `name` and `layer`.

Parameters

layer1 layer2 ...	Specifies the layer(s) and/or via(s). For layer names, specify the number—for example 3. For via names specify the lowercase character v followed by the via number—for example, v3.
llx	Specifies the lower left x coordinate of the bounding box.
lly	Specifies the lower left y coordinate of the bounding box.
`name`	Specifies the name of the routing blockage.
urx	Specifies the upper right x coordinate of the bounding box.
ury	Specifies the upper right y coordinate of the bounding box.

Examples

- The following command selects the routing blockage `blk1` that lies in the bounding box with the coordinates `17`, `223`, `1117`, and `1223`. The commands selects the routing blockage on layer `1` and layer `2`.

  ```
  selectRouteBlk 17 223 1117 1223 blk1 {1 2}
  ```

- The following command selects the routing blockage `blk1` on layer `3` and `4`.

  ```
  selectRouteBlk blk1 {3 4}
  ```

- The following command selects the routing blockage `blk2` that lies in the bounding box with the coordinates 17, 223, 1117, and 1223. The commands selects the routing blockage on layer1 and via 2.

  ```
  selectRouteBlk 17 223 1117 1223 blk2 {1 v2}
  ```

setBlockPlacementStatus

```
setBlockPlacementStatus
     -allHardMacros | -allPtnBlks | -allBlackBoxes | -name instName
     -status {unplaced | fixed | placed | cover}
```

Changes the placement attributes status for all hard macros, all partition blocks, all blackboxes, or selected instances.

You can use this command after running the `modulePlace` command. This allows you to fix all blocks so they are not moved by `placeDesign`.

Parameters

`-allHardMacros` Applies the status option only to all hard macros.

`-allPtnBlocks` Applies the status option only to all partition blocks.

`-allBlackBoxes` Applies the status option only to all blackboxes.

`-name` *`instName`* Applies the status options to the specified instance(s). You can use a wildcard to specify multiple instances.

`-status {unplaced | fixed | placed | cover}`

Specifies the change in placement status for instances.

- `cover` Specifies the change in placement status to be a *cover* status.
- `fixed` Specifies the change in placement status to a *fixed* status.
- `placed` Specifies the change in placement status to a *placed* status.
- `unplaced` Specifies the change in placement status to an *unplaced* status.

Example

The following commands set all hard macros and blackboxes to preplaced (`FIXED` status in the DEF file) after a `modulePlace` run. By setting this status, `placeDesign` can run without moving these preplaced blocks.

```
modulePlace
setBlockPlacementStatus -allHardMacros -status preplaced
setBlockPlacementStatus -allBlackBoxes -status preplaced
```

setBottomIoPadOrient

`setBottomIoPadOrient` *`orient`*

Changes the orientation for the bottom pad (South side). Changing the bottom pad orientation changes the pad relationship of the East, North, and West side pads.

Parameters

`orient`	Specifies the orientation. The possible orientation values are `R0`, `R90`, `R180`, and `R270`.

Example

The following command changes the orientation of the bottom I/O pad instances from R0 to `R270` for the bottom pad (South side):

```
setBottomIoPadOrient R270
```

As a result, the respective orientation of the I/O pad instances change on the East (`R270` to `R180`), North (`R180` to `R90`), and West (`R90` to `R0`) sides.

setDrawView

```
setDrawView {fplan | amoeba | place}
```

Sets the design view in the design display area.

To see the current view use `getDrawView`.

You can use this command after importing or restoring the design.

Parameters

`amoeba`	Sets the view to Amoeba view. This view displays the outline of the modules and submodules after placement, showing physical locality of the module.
`fplan`	Sets the view to floorplan view. This view displays the hierarchical module and block guides, connection flight lines, and floorplan objects, including block placement, and power and ground nets.
`place`	Sets the view to physical view. This view displays the detailed placements of the module's blocks, standard cells, nets, and interconnects.

setFixedBlockSize

```
setFixedBlockSize
     {* | listOfInstances}
```

Specifies that during floorplan resizing, the size of the specified modules and/or blackboxes should not change.

Use this command before specifying the floorplan.

Parameters

*	Specifies that the size of *all* the modules and/or blackboxes should not change during floorplan resizing.
listOfInstances	Specifies that the size of the *specified* modules and/or blackboxes should not change during floorplan resizing. Provide a space-separated list of instances.

Example

The following command specifies that the size of `instanceA` and `instanceB` should not change during floorplan resizing.

```
setFixedBlockSize instanceA instanceB
```

setFlipping

```
setFlipping
     n | s | f
```

Specifies the orientation of the bottom row in the core area. The command recreates all the floorplan rows.

Parameters

f	Specifies that the first row flips from the bottom up.
n	Specifies no row flipping.
s	Specifies that the second row flips from the bottom up.

setFPlanRowSpacingAndType

```
setFPlanRowSpacingAndType
     spacingValue
     1 | 2
```

Specifies the standard row spacing values and row pattern type.

Parameters

spacingValue	Specifies the standard row spacing, in micrometers. This must be a positive value. By default, this value is zero.
1 | 2	Specify 1 for every row, or 2 for every other row to apply the spacing value.

Example

The following command specifies a row spacing value of 1.1 micrometers for every other row:

```
setFPlanRowSpacingAndType 1.1 2
```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Standard Row Spacing"

setInstGroupPhyHier

`setInstGroupPhyHier` *`groupName`*

Sets a pre-existing instance group into a physical hierarchy where you can generate a corresponding netlist.

You can use this command after creating a group name.

Parameters

`groupName`	Specifies the name of the created group.

Example

The following commands create and set the instance group `adder1` as a physical hierarchy:

```
createInstGroup adder1
setInstGroupPhyHier adder1
```

setIoFlowFlag

```
setIoFlowFlag
     [-help]
     {0 |1}
```

Sets the flag to use I/O row flow for pad placement.

To view the current flag setting, use the `getIoFlowFlag` command.

Parameters

`{0 | 1}`	Specify `1`, to use the I/O row flow. Specify `0`, to use the normal I/O flow.
`-help`	Outputs a brief description that includes type and default information for each `setIoFlowFlag` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man setIoFlowFlag`.

Related Topics

- "Floorplan Menu" chapter of the *Encounter Menu Reference*
  - "Create I/O Row"
- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Performing I/O Row Based Pad Placement"
- "Data Preparation" chapter of the *Encounter User Guide*
  - "Creating an I/O Assignment Flow"

setIoRowMargin

```
setIoRowMargin
     { n | w | s | e }
     rowNumber
     marginDistance
```

Sets the distance from the die boundary edge to the I/O row starting edge location. You can use this command for multiple I/O rows.

Parameters

`{ n | w | s | e }`	Specifies the side of the I/O row margin, where `n` is the North side, `w` is the West side, `s` is the South side, and `e` is the East side.
`marginDistance`	Specifies the distance between the die boundary edge to the I/O row edge, in microns.

`rowNumber`	Specifies the I/O row number. This value can be up to 8 rows. You can also use this value in conjunction with the `placePADIO` and `placePIO -optIOs` commands to specify the number of rows for placing peripheral I/O cells for the CLASS PAD and CLASS PAD AREAIO flows respectively.
	■ To create and optimize multiple I/O Rows in a CLASS PAD flow, use the following commands:
	`setIoRowMargin` *`rowNumber`*
	`placePADIO`
	■ To create and optimize multiple I/O Rows in a CLASS PAD AREAIO flow, use the following commands:
	`setIoRowMargin` *`rowNumber`*
	`placePIO -optIOs`

Example

The following command sets the North side of the second I/O row starting edge value at 300 µm from the die boundary edge to the second I/O row edge:

```
setIoRowMargin n 2 300
```

setNdrSpacing

```
setNdrSpacing
     [-layer layerName]
     spacing | -hardSpacing {0 | 1}
```

Sets the spacing of the metal layer specified.

Parameters

`-layer` *`layerName`*	Specifies the name of the metal layer whose spacing is to be defined.
`spacing`	Specifies the spacing, in micrometers, of the metal layer.
`-hardSpacing {0 | 1}`	If set to 1, the specified spacing is treated as “hard” spacing rule.

Example

The following command sets the spacing of the metal layer M2 to 0.8 micrometers.

```
setNdrSpacing -layer M2 0.8
```

Related Topics

- “Using the NanoRoute router” chapter in the *Encounter User Guide*
  - “Using Non-Default Rules”

setNdrWidth

```
setNdrWidth
     [-layer layerName]
     width
```

Sets the width of the metal layer in the temporary non-default rule structure.

Parameters

`-layer layerName`	Specifies the name of the routing metal layer. **Note:** If you omit this parameter, the width that you specify is applied to all the routing metal layers.
`width`	Specifies the width, in micrometers, of the routing metal layer.

Example

The following command sets the width of the metal layer `M2` to 0.46 micrometers in the LEF file.

```
setNdrWidth -layer M2 0.46
```

Related Topics

- "Using the NanoRoute router" chapter in the *Encounter User Guide*
  - "Using Non-Default Rules"

setObjFPlanBox

```
setObjFPlanBox
     objectType
     objectName
     llx lly urx ury
```

Defines the bounding box of a specified object, even outside the core boundary.

Parameters

objectName	Specifies the name of the object for *objectType*.
objectType	Specifies the type of object for *objectName*. This can be any of the following: ■ Bump ■ Cell ■ Group ■ Instance ■ I/O cell ■ I/O pin ■ Layershape ■ Module ■ Net ■ Partition cut ■ Partition feedthrough ■ Partition pin block ■ Pin ■ Pin guide ■ Row cluster ■ Standard row
llx	Specifies the lower left x coordinate of the object.
lly	Specifies the lower left y coordinate of the object.

urx	Specifies the upper right x coordinate of the object.
ury	Specifies the upper right y coordinate of the object.

Example

The following command specifies a bounding box for module abc at a lower left x coordinate of 100.00, a lower left y coordinate of 100.00, and upper right x coordinate of 400.00, and an upper right y coordinate of 545.00:

```
setObjFPlanBox Module abc 100.00 100.00 400.00 545.00
```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Defining the Bounding Box"

setObjFPlanBoxList

```
setObjFPlanBoxList
     {Cell | Group | Instance | Layershape | Module | Net | Pin}
     objectName
     llx1 lly1 urx1 ury1
     llx2 lly2 urx2 ury2
     ...
     llxn llyn urxn uryn
```

Defines the rectilinear shape of an object even if the object lies outside the core area. A rectilinear object comprises of two or more boxes.

Parameters

objectName	Specifies the name of the object for object type.
{Cell | Group | Instance | Layershape | Module | Net | Pin}	Specifies the type of object for *objectName*.
llx1	Specifies the lower left x coordinate of the first box in the box list.
lly1	Specifies the lower left y coordinate of the first box in the box list.
llx2	Specifies the lower left x coordinate of the second box in the box list.
lly2	Specifies the lower left y coordinate of the second box in the box list.
urx1	Specifies the upper right x coordinate of the first box in the box list.
ury1	Specifies the upper right y coordinate of the first box in the box list.
urx2	Specifies the upper right x coordinate of the second box in the box list.
ury2	Specifies the upper right y coordinate of the second box in the box list.
llxn	Specifies the lower left x coordinate of the n^{th} box in the box list.
llyn	Specifies the lower left y coordinate of the n^{th} box in the box list.
urxn	Specifies the upper right x coordinate of the n^{th} box in the box list.
uryn	Specifies the upper right y coordinate of the n^{th} box in the box list.

Example

The following command defines a rectilinear boundary for module `xyz`. The rectilinear boundary is made up of two bounding boxes:

1. (371.46, 537.60) (696.96, 754.35)
2. (412.5, 754.32) (696.96, 920.64)

```
setObjFPlanBoxList Module xyz 371.46 537.60 696.96 754.35 412.5 754.32 696.96
920.64
```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Defining the Bounding Box"

setObjFPlanPolygon

```
setObjFPlanPolygon
     objectType {cell | group | instance | layershape | module | net | pin}
     objectName
     {x1 y1 x2 y2 x3 y3 ...}
```

Specifies a rectilinear object with polygon coordinates. The rectilinear object can take any shape depending on the number of x and y coordinates you specify.

Parameters

objectName	Specifies the name of the object for *objectType*.
objectType	Specifies the type of object for *objectName*. This can be: ■ Cell ■ Group ■ Instance ■ Layer shape ■ Module ■ Net ■ Pin
x1 y1 x2 y2 ...	Specifies the corner coordinates of the rectilinear object.

Example

The following command defines a rectilinear object—module xyz with five pairs of corner coordinates.

```
setObjFPlanPolygon module xyz 371.46 537.60 696.96 754.35 412.5 754.32 696.96
920.64 696.96 537.60
```

setPlanDesignMode

```
setPlanDesignMode
     [-help]
     [-reset]
     [-abSpacingX x_micron]
     [-abSpacingY y_micron]
     [-autoPowerPlan total_power_in_mW]
     [-boundaryPlace {true | false}]
     [-congAware {true | false}]
     [-fixPlacedMacros {true | false}]
     [-groupHardMacro {true | false}]
     [-groupIOLogic {true | false}]
     [-handleFlat one_character]
     [-keepGuide {true | false}]
     [-maxDistToGuide ratio]
     [-noColorize {true | false}]
     [-numSeed number]
     [-powerAware {true | false}]
     [-seedSize inst_per_seed]
     [-setSeedHierLevel level]
     [-spacingX x_micron]
     [-spacingY y_micron]
     [-useExistingPowerRail {true | false}]
     [-util module_util]
```

Sets the Masterplan automatic floorplanner global parameters. Parameters specified with the `setPlanDesignMode` command are then used whenever you run the `planDesign` command to create an initial floorplan.

Use the `getPlanDesignMode` command to return the current settings for the `setPlanDesignMode` command.

Typically, you run the `planDesign` command using the default automatic seed selection. By default, Masterplan selects seeds using the following methods in order:

- Chooses modules that fit an internally calculated size range
- Chooses large hard macros as seeds
- Groups small modules or single standard cells to fit an internally calculated size range

However, you can force automatic seed selection to favor a certain constraint during the seed selection process by specifying one of the `-setSeedHierLevel`, `-numSeed`, and `-seedSize` parameters.

You can use the `setPlanDesignMode` command after importing a design and before using the `planDesign` command.

Parameters

`-abSpacingX` *`x_micron`*

Specifies the absolute horizontal spacing between adjacent hard macros. You can use this parameter to add space in order to decrease congestion between hard macros.

Type: Float

Default: `-1` (Spacing is (25 + *num_pin_on_edge*/8) x *M2_pitch*)

`-abSpacingY` *`y_micron`*

Specifies the absolute vertical spacing between adjacent hard macros. You can use this parameter to add space in order to decrease congestion between hard macros.

Type: Float

Default: `-1` (Spacing is (25 + *num_pin_on_edge*/8) x *M2_pitch*)

`-autoPowerPlan` *`total_power_in_mW`*

Calls the automatic power planner to generate a generic power plan after macro placement, based on the specified total average power value. You can use this power plan for estimating congestion, or performing basic power analysis.

The following conditions exist for this parameter:

- To use `-autoPowerPlan` for a chip design, the power pads must already be placed in the design.
- The global power net connection must be defined.
- The `-autoPowerPlan` parameter uses the default template for power planning.

Note: The `-autoPowerPlan` and `-useExistingPowerRail` parameters are mutually exclusive: If you specify `-autoPowerPlan`, you cannot specify `-useExistingPowerRail true`.

Default: `0`

`-boundaryPlace {true | false}`

Places macros along the core boundary for chip and block designs. Macros are placed along the fence boundary for hierarchical floorplans. Preplaced hard macro boundaries are also used, if possible.

Note: Specifying this parameter can disable the `-maxDistToGuide` parameter. Seed locations might also be ignored.

Default: `false`

`-congAware {true | false}`

Estimates the congestion for the floorplan after macro placement.

If congestion (the Trial Route estimated Gcell overflow percentage) is greater than 0.5 percent in either the X or Y direction, Masterplan attempts to resolve it by refining the macro placement (such as changing the spacing, orientation, or grouping). Masterplan does not touch user-preplaced macros during this refinement. After re-estimating the congestion, if the congestion improved, Masterplan keeps the new floorplan. If congestion did not improve, Masterplan restores the previous floorplan.

If you specify the `-autoPowerPlan` parameter with `-congAware true`, the automatic power planner generates a power plan that Masterplan takes into consideration when estimating and resolving congestion.

If you specify `-useExistingPowerRail true` with `-congAware true`, Masterplan uses the existing power routing when estimating and resolving congestion.

Note: For better accuracy, it is recommended that you have power information available, to use this feature.

Default: `false`

`-fixPlacedMacros {true | false}`

Marks all placed macros as `FIXED` after running `planDesign`. Marking the macros as `FIXED` prevents them from accidentally being moved during `placeDesign`.

Default: `false`

`-groupHardMacro {true | false}`

Controls the selection of hierarchical instances as seeds, based on hard macro distribution. When set to `true`, if hierarchical instances at a certain hierarchy level contain the same types of macros, Masterplan picks their common parent hierarchical instance as a seed.

Default: `false`

`-groupIOLogic {true | false}`

Creates instance group seeds by traversing the logic from the I/O pads (or pins) to the first level of sequential elements. When set to `true`, Masterplan creates a maximum of eight instance group seeds (two on each side), depending on the design size.

Note: You must have a timing library specified in order to set this parameter to `true`.

Default: `false`

`-handleFlat` *`one_character`*

Forces automatic seed selection to rebuild the original logic hierarchy, and create instance group seeds based on a flattened netlist, and the specified hierarchy delimiter. For example:

```
setPlanDesignMode -handleFlat /
```

Default: /

`-help`

Outputs a brief description that includes type and default information for each `setPlanDesignMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setPlanDesignMode`.

`-keepGuide {true | false}`

Retains Masterplan-generated module guides after the floorplan has been created.

Default: `false`

`-maxDistToGuide` *`ratio`*

Restricts macro placement to an area that is relative to the macro's associated guide boundary.

Type: Real number, between `0` and `1`

Default: `0.0625` (1/16)

`-noColorize {true | false}`

Uses the default Encounter color scheme to color module guides and macros in the resulting floorplan, instead of the Masterplan color scheme.

Default: `false`

`-numSeed` *`number`*

Forces automatic seed selection to select approximately the specified number of seeds. For most designs, the optimal number of seeds to select is between 20 to 50. You should not select fewer than 10 seeds, or more than 100.

The total number of seeds actually selected by the tool can be within a range of plus or minus 5 of the specified number.

Set this parameter to `true` only when you want to bias the seed selection process to favor seed number when selecting seeds.

Note: If you set this parameter to `true`, you cannot specify `-seedSize true` or `-setSeedHierLevel true`.

Type: Integer

Default: `20`

`-powerAware {true | false}`

Performs IR drop analysis after macro placement, and displays an IR drop map.

If you specify the `-autoPowerPlan` parameter with `-powerAware true`, the automatic power planner generates a power plan that Masterplan uses for IR drop analysis.

If you specify `-useExistingPowerRail true` with `-powerAware true`, Masterplan uses the existing power routing for IR drop analysis.

The following conditions exist for this parameter:

- For better accuracy, it is recommended that you have power information available, to use this feature.
- Masterplan currently analyzes only one power net by default (the one with the most instance connections).
- The default threshold for the IR drop map is 5 percent of the main power supply voltage.

Default: `false`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setPlanDesignMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-seedSize` *`inst_per_seed`*

Forces automatic seed selection to use the specified size (in terms of area) as the basic seed size when grouping instances.

Set this parameter to `true` only when you want to bias the seed selection process to favor seed size when selecting seeds.

Note: If you set this parameter to `true`, you cannot specify `-numSeed true` or `-setSeedHierLevel true`.

Type: Integer

Default: `128`

`-setSeedHierLevel` *`level`*

Forces automatic seed selection to select all hierarchical instances at the specified logic level as seeds. The top design cell is at hierarchy level 0.

Set this parameter to `true` only when you want to bias the seed selection process to favor hierarchy when selecting seeds. additionally, your hierarchy levels should be very well defined and organized.

Note: If you set this parameter to `true`, you cannot specify `-numSeed true` or `-seedSize true`.

Type: Integer

Default: `1`

`-spacingX` *`x_micron`* Specifies extra horizontal spacing to add to the default spacing between adjacent hard macros. You can use this parameter to add space in order to decrease congestion between hard macros.

Type: Float

Default: `-1` (Spacing is (25 + *num_pin_on_edge*/8) x *M2_pitch)*

`-spacingY` *`y_micron`* Specifies extra vertical spacing to add to the default spacing between adjacent hard macros. You can use this parameter to add space in order to decrease congestion between hard macros.

Type: Float

Default: `-1` (Spacing is (25 + *num_pin_on_edge*/8) x *M2_pitch*)

`-useExistingPowerRail {true | false}`

Uses the existing power routing in the design for estimating congestion, or performing basic power analysis.

Note: The `-autoPowerPlan` and `-useExistingPowerRail` parameters are mutually exclusive: If you specify `-autoPowerPlan`, you cannot specify `-useExistingPowerRail true`.

Default: `true`

`-util` *`module_util`*

Specifies the target utilization for generated floorplan guides. You can use this parameter to decrease congestion between fences during hierarchical floorplanning.

Type: Float, between 0.000000 and 1.000000

Default: `0.75`

In the fence generation flow with utilization set in `setPlanDesignMode`, the regular fence without any macro honors global utilization value. For fence with macro, the automatic fence generator uses lower utilization for fence (bigger in area), to make macro placement easier to legalize, and tries to get the utilization closer to global utilization.

Any fence specific utilization setting in the constraint file overrides the global utilization setting for that fence.

Examples

- The following command places macros along the core boundary for chip and block designs when running `planDesign`:

  ```
  setPlanDesignMode -boundaryPlace true
  ```

- The following command instructs Masterplan to estimate and resolve congestion based on the existing power routing in the design:

  ```
  setPlanDesignMode -congAware true -useExistingPowerRail true
  ```

- The following command resets the `-boundaryPlace` parameter to its default value:

  ```
  setPlanDesignMode -reset -boundaryPlace
  ```

- The following command resets all `setPlanDesignMode` parameters to their default values:

  ```
  setPlanDesignMode -reset
  ```

setResizeLine

```
setResizeLine
     {-direction {H|V} (x1 y1) (x2 y2) ... [-width widthInMicrons]} |
     -clearAll
```

Sets the resize lines for shift-based floorplan resize option of the `resizeFP` command. The resize lines can be non-continuous though they must be orthogonal. If resize lines are specified, the floorplan will be resized between the area specified by the resize lines (for shift-based floorplan resize).

To shrink or expand the floorplan in the horizontal direction, specify a vertical resize line. To shrink or expand the floorplan in the vertical direction, specify a horizontal resize line.

Each resize line can have multiple segments (for example, horizontal, vertical, and again horizontal) but it can only be applied to one direction. If you want to shrink or expand the floorplan in both directions, specify resize lines for each direction.

You can create multiple resize lines. Resize lines in the same direction (horizontal or vertical) can overlap. The resize lines are removed once the `resizeFP` command is run.

You can use this command after importing the design.

Parameters

`-clearAll`
: Clears all resize lines.

`-direction {H|V}`
: Specifies whether the resize line is horizontal (`H`) or vertical (`V`)

`(x1 y1) (x2 y2) ...`
: Specifies the x and y coordinates of the resize line.

  You can specify multiple resize lines using the coordinates. You can also specify non continuous lines by specifying a * for x or y coordinate. For example, you can specify (x1 y1) (x2 *) (x3 y3) (x4 *). In such cases, the missing coordinates will be automatically derived.

`-width widthInMicrons`
: Specifies the width of the resize line in microns. A positive value expands the floorplan and a negative value shrinks the floorplan.

Examples

- The following command sets a resize line with a width of 20 microns in the horizontal direction. As the width has a positive value, the floorplan will expand. The resize line is horizontal and will, therefore, expand the floorplan in a vertical direction.

  ```
  setResizeLine -direction H -width 20 (1924 3476) (3807 3476)
  ```

- The following command sets a resize line with a width of -20 microns in the horizontal direction. As the width has a negative value, the floorplan will shrink. The resize line is horizontal and will, therefore, shrink the floorplan in a vertical direction.

  ```
  setResizeLine -direction H -width -20 (1924 3476) (3807 3476)
  ```

- The following command sets a resize line with a width of 20 microns in the vertical direction. As the width has a positive value, the floorplan will expand. The resize line is vertical and will, therefore, expand the floorplan in a horizontal direction.

  ```
  setResizeLine -direction V -width 20 (2761 2215) (2761 3810)
  ```

- The following command sets a resize line with a width of -20 microns in the vertical direction. As the width has a negative value, the floorplan will shrink. The resize line is vertical and will, therefore, shrink the floorplan in a horizontal direction.

  ```
  setResizeLine -direction V -width -20 (2761 2215) (2761 3810)
  ```

- The following command sets a resize line with a width of -20 microns in the vertical direction. As a * has been specified for one of the coordinates, the value of that coordinate will be automatically derived.

  ```
  setResizeLine -direction V -width -20 (2761 *) (2761 3810)
  ```

- The following command creates a resize line with multiple segments.

  ```
  setResizeLine -direction V -width -20 (2643 1744) (2643 3411) (3382 3411)
  (3382 4117)
  ```

setRouteBlkDefaultLayer

```
setRouteBlkDefaultLayer
     {-layer {layerId... | all} | -cutLayer {layerId... | all}
     | -allMetalCut}
```

Specifies the default layers for routing blockages, for both metal layers and cut layers. The default layers set with this command are applicable for all routing blockages subsequently created using the GUI or the `createRouteBlk` command.

Note: If any layers are specified with the `createRouteBlk` command, the layers specified with the `setRouteBlkDefaultLayer` command are ignored.

Use this command before creating routing blockages.

Parameters

`allMetalCut` Specifies that default routing blockages will be created for all metal layers and all cut layers.

`-layer {`*`layer1d...`*` | all}`

- `-layer` *`layerId...`* specifies that default routing blockages will be created for the specified metal layers.
- `-layer` *`all`* specifies that default routing blockages will be created for all metal layers.

`-cutLayer {`*`layerId...`*` | all}`

- `-cutLayer` *`layerId...`* specifies that default routing blockages will be created for the specified cut layers.
- `-cutLayer` *`all`* specifies that default routing blockages will be created for all cut layers.

Example

The following command sets default routing blockages for the metal layers1, 2, and 3 and the cut layers 3 and 5.

```
setRouteBlkDefaultLayer -layer 1 2 3 -cutLayer 3 5
```

setSelectedDensityArea

```
setSelectedDensityArea
     x1 y1 x2 y2
     density
     name
     type
     attribute
```

Specifies the attributes for the selected density screen area(s).

Parameters

x1 y1 x2 y2	Specifies the coordinates of the density screen area.
density	Specifies the standard cell placement density percentage. The valid values are in increments of 5: 0, 5, 10, ...100. For example, setting a density percentage of 75 percent means that up to 75 percent of the defined density area will be used for placement. **Note:** If you enter any number that is not valid, the percentage defaults to the nearest valid value.
name	Specifies the name of the density screen area.
type	Specifies the hardness of the density screen area. (`Partial` or `Soft`)
attribute	Specifies the relationship of the density screen area to the design. ■ `Instance` The density screen area is associated with the current design. ■ `Pushdown` The density screen area is pushed down from the hierarchy. ■ `Undefined` The relationship of the density screen area to the design is undefined. *Default*: `Undefined`.

Example

The following command sets a soft density screen area, `softDensity2`, of the current design in the specified design area:

```
setSelectedDensityArea 100.0 100.0 200.0 200.0 75 softDensity2 Soft Instance
```

setSelectedObstruct

```
setSelectedObstruct
     x1 y1 x2 y2
     obstructName
     attribute
```

Sets the placement blockage attributes, like the area, name, and type, for the selected placement blockage (only).

Parameters

x1 y1 x2 y2	Specifies the coordinates of the placement blockage area.
obstructName	Specifies the name of the placement blockage.
attribute	Specifies the type of placement blocakge. ■ `Instance` The placement blocakge is associated with the current design. ■ `Pushdown` The placement blocakge is pushed down from the hierarchy. ■ `Undefined` The type of placement blockage is undefined. *Default*: `Undefined`.

setSelectedRouteBlk

```
setSelectedRouteBlk
     llx lly urx ury
     name
     {layer1 layer2 ...}
     [-attr {instance | pushdown | slots | fills | exceptpgnet | undefined}]
     [-spacing value | -designRuleWidth value]
```

Sets a routing blockage for an attribute of the selected routing blockage.

Parameters

`-attr {instance | pushdown | slots | fills | exceptpgnet | undefined}`

Specifies the attribute of the routing blockage. This can be:

- `instance`
- `pushdown`
- `slots`
- `fills`
- `exceptpgnet`
- `undefined`

Default: `undefined`.

Note: This is an optional parameter.

`-designRuleWidth value`

Specifies the effective width of the routing blockage. The value is a real number greater than 0.

layer1 layer2 ...

Specifies the layer(s) and/or via(s). For layer names, specify the number—for example 3. For via names, specify the uppercase character V followed by the via number—for example, V3.

llx

Specifies the lower left x coordinate of the bounding box.

lly

Specifies the lower left y coordinate of the bounding box.

name

Specifies the name of the routing blockage.

urx	Specifies the upper right x coordinate of the bounding box.
ury	Specifies the upper right y coordinate of the bounding box.
-spacing *value*	Specifies the minimum spacing between layers within the routing blockage. The value is a real number greater than or equal to 0.

Examples

- The following command sets the routing blockage for fills in routing blockage blk2 that lies in the bounding box with the coordinates 17, 223, 1117, and 1223, on metal layer1 and via 2.

```
setSelectedRouteBlk 17 223 1117 1223 blk2 {1 V2} [fills]
```

setSelectedStripBoxShape

```
setSelectedStripBoxShape
     index
     value
```

Sets the shape of the selected strip box.

Parameters

index

Specifies which selected strip box shape needs to be set.

You can specify one of the following:

- `all`

  Specifies that the shape of all the selected strip boxes must be set.

- `integer`

  Specifies that the shape of only the selected strip box must be set.

  For example, if you have 4 strip boxes selected, an index value, 2, specifies that the shape of the third selected strip box must be set.

value Specifies the shape of the selected strip box.

You can specify one of the following values:

- None
- RING
- STRIPE
- FOLLOWPIN
- IOWIRE
- COREWIRE
- BLOCKWIRE
- FILLWIRE
- PADRING
- DRCFILL
- FILLWIREOPC

Examples

1. If you selected 5 strip boxes, the following command sets the shape of the fifth strip box to RING.

   ```
   setSelectedStripBoxShape 4 RING
   ```

2. If you selected 4 strip boxes, the following command sets the shape of all the 4 strip boxes to RING.

   ```
   setSelectedStripBoxShape all RING
   ```

setSelectedStripBoxState

```
setSelectedStripBoxState
     index
     value
```

Sets the state of the selected strip box.

Parameters

index

Specifies which selected strip box state needs to be set.

You can specify one of the following:

- `all`

  Specifies that the state of all the selected strip boxes must be set.

- `integer`

  Specifies that the state of only the selected strip box must be set.

  For example, if you have 4 strip boxes selected, an index value, 2, specifies that the state of the third selected strip box must be set.

value

Specifies the state of the selected strip box.

You can specify one of the following values:

- `ROUTED`
- `FIXED`
- `COVER`
- `SHIELD`

Examples

1. If you selected 5 strip boxes, the following command sets the state of the fifth strip box to `FIXED`.

   ```
   setSelectedStripBoxState 4 FIXED
   ```

2. If you selected 4 strip boxes, the following command sets the state of all the 4 strip boxes to `ROUTER`.

```
setSelectedStripBoxState all ROUTED
```

shiftInst

```
shiftInst
     -side {right | left | up | down}
     -distance value
```

Shifts instances vertically or horizontally by a specified distance.

You can use this command after selecting one or more instances in the design display area.

Parameters

`-distance` *`value`*	Specifies the shift distance value, in micrometers.	
`-side`	Specifies the vertical and horizontal instance shift for the selected instances. Choose one of the following:	
	`right`	Specifies a right shift.
	`left`	Specifies a left shift.
	`up`	Specifies an upward shift.
	`down`	Specifies a downward shift.

Example

The following command shifts highlighted instances `instA` and `instB` upwards 5 micrometers:

```
shiftInst -side up -distance 5
```

snapFPlan

```
snapFPlan
     -all | -selected |
     {[-guide] [-block] [-stdCell] [-ioPad] [-ptnCore] [-pinGuide]
     [-routeBlk][-pin][-pinBlk]}
```

Snaps floorplan objects to the grid.

You can use this command after importing the design.

Parameters

`-all`	Specifies the snapping of all supported object types.
`-block`	Specifies the snapping of floorplan blocks.
`-guide`	Specifies the snapping of floorplan guides, regions, and fences.
`-iopad`	Specifies the snapping of I/O pads.
`-pin`	Specifies the snapping of pins.
`-pinBlk`	Specifies the snapping of pin blockages.
`-pinGuide`	Specifies the snapping of pin guides.
`-ptnCore`	Specifies the snapping of the partition core.
`-routeBlk`	Specifies the snapping of routing blockages.
`-selected`	Specifies the snapping of all supported object types that are selected in the design display area.

snapFPlanIO

```
snapFPlanIO
     [-selected]
     [-userGrid | -toIoRow [-overlapRowOnly]]
```

Snaps I/O cells to a user-defined grid.

Use this command after importing the design.

Note: You specfiy the user-defined grid in the *Preferences - Floorplan* form in the Design menu.

Parameters

`-overlapRowOnly`	Snaps the I/O cell only onto an overlapping I/O row. If an I/O row is not found, the command is not executed.
`-selected`	Applies to the selected I/Os only. *Default*: snaps to all I/O cells.
`-toIoRow`	Snaps the I/O cell onto an overlapping I/O row first. If an I/O row is not found, places the I/O cell in any row having the same site.
`-userGrid`	Specifies the snapping of I/O pads to user-defined grids. *Default*: snaps I/O to manufacture grid.

spaceInst

```
spaceInst
     -fixSide {right | left | center | top | bottom | middle | distHorizontal |
     distVertical}
     -space value
```

Spaces instances horizontally or vertically by a specified distance value, and can evenly distribute the spacing horizontally or vertically between three or more instances.

Note: To space I/O cells, use the `spaceIoInst` command.

You can use this command after selecting two or more instances in the design display area.

Parameters

`-fixSide` Specifies the vertical and horizontal instance spacing. Choose one of the following:

`right`	Specifies that the right most instance is fixed and all other instances are shifted horizontally to the left, maintaining space defined in the `-spacing` option.
`left`	Specifies that the left most instance is fixed and all other instances are shifted horizontally to the right, maintaining space defined in the `-spacing` option.
`center`	Specifies that the center most instance is fixed and all other instances are shifted horizontally around the center, maintaining space defined in the `-spacing` option.
`top`	Specifies that the top most instance is fixed and all other instances are shifted vertically to the bottom, maintaining space defined in the `-spacing` option.
`bottom`	Specifies that the bottom most instance is fixed and all other instances are shifted vertically to the top, maintaining space defined in the `-spacing` option.

	`middle`	Specifies that the middle instance is fixed and all other instances are shifted vertically, maintaining space defined in the `-spacing` option.
	`distHorizontal`	Specifies an even horizontal distribution.
	`distVertical`	Specifies an even vertical distribution.
`-space` *`value`*	Specifies the spacing value, in micrometers.	

Example

The following command fixes the bottom-most highlighted instance and moves all the other instances vertically to the top, spacing the instances 5 micrometers apart from each other:

```
spaceInst -fixSide bottom -space 5
```

spaceIoInst

```
spaceIoInst
     [-fixSide left | right | top | bottom | horDistribute | vertDistribute]
     [-space value]
```

Spaces the selected I/O cells horizontally or vertically by a specified distance value, and can evenly distribute the spacing horizontally or vertically between two or more I/O cells.

You can use this command after selecting two or more instances in the design display area.

Note: This command supports I/O row flow. For more information, see Performing I/O Row Based Pad Placement in "Floorplanning the Design" chapter of the *Encounter User Guide.*

Parameters

`-fixSide`	Specifies the vertical and horizontal instance spacing. Choose one of the following:	
	`right`	Specifies horizontal right spacing.
	`left`	Specifies horizontal left spacing.
	`top`	Specifies vertical top spacing.
	`bottom`	Specifies vertical bottom spacing.
	`horDistribute`	Specifies an even horizontal distribution.
	`vertDistribute`	Specifies an even vertical distribution.
	Default: The default value is `left`.	
`-space value`	Specifies the spacing value, in micrometers.	
	Default: The default value is `0`.	

Example

The following command aligns the right-most highlighted I/O cell horizontally with a spacing of 5 micrometers:

```
spaceIoInst -fixSide right -space 5
```

specifyBlackBlob

```
specifyBlackBlob
     -cell <cellName>
     {-area totalArea
     | {-gateArea gateArea includesMacroArea {0|1}}
     | {-gateCount NumberOfGates areaPerGate includesMacroArea {0|1}}}
     [-macroList {macroName [count]}...]
     [-placePorosity percentValue]
     [-targetUtil percentValue]
     [-routePorosity {layerName percentValue}...]
```

Creates a blackblob with the specified parameters.

You can use this command after importing the design.

Important

After you have specified the blackblobs in the design, run the `elaborateBlackBlob` command to instantiate the hard macros in the blackblob and complete the virtual connections for super flaked cells and the embedded macros. Typically, you would run the `elaborateBlackBlob` command once after you have specified the blackblobs in the design with the `specifyBlackBlob` command. If you specify blackblobs through the Specify Blackblob GUI, the `elaborateBlackBlob` command will automatically be invoked when you click the Apply or the OK button.

Parameters

`-area totalArea`

Specifies the total area of the blackblob in square microns. If the -targetUtil is not specified then the gate area (or bloblet area) is equal to the total blob area. If the targetUtil value is specified then the gate area is calculated based on the total area of the blackblob and the user-specified targetUtil value.

For example, the following command specifies the total area and the target utilization percentage of the blackblob cell `tdsp_core`.

```
specifyBlackBlob -cell tdsp_core -area 125 -targetUtil 80
```

The calculated gate area = (total area * targetUtil) / 100
= (125 * 80) / 100 = 100

`-cell cellName`

Specifies the name of the blackblob cell.

`-gateArea`

`gateArea` Specifies the total area of the gate. While using this parameter, the gate area is fixed and the total area is adjusted based on the targetUtil.

For example, the following command specifies the gate area and the target utilization percentage of the blackblob cell `tdsp_core`.

```
specifyBlackBlob -cell tdsp_core
-gateArea 100 -targetUtil 80
```

The calculated total blackblob area
= (gate area * 100) / targetUtil
= (100*100) / 80 = 125

If the gate count is specified, the gate area is calculated based on the gate count and the area per gate.

`includesMacroArea {0|1}`

A value of `0` specifies that the gate area does not include the gate area of any hard macro(s) present in the design.

A value of `1` specifies that the gate area includes the gate area of any hard macro(s) present in the design.

Default: The default value is `0`. That is, the gate area does not include the gate area of any hard macro(s) present in the design

`-gateCount`

`gateCount` Specifies the number of gates.

`areaPerGate` Specifies the area per gate.

`NumberOfGates` Specifies the total number of gates.

`includesMacroArea {0|1}`

Specifies whether the gate area calculations based on the gate count should include the area of any specified hard macro(s).

A value of `1` specifies that the gate area will be calculated as follows:

```
(gate count) * (area per gate)
```

A value of `0` specifies that the gate area will be calculated as follows:

```
{(gate count) * (area per gate)} + area
of specified macro(s)
```

Default: The default value is `0`.

`-macroList {`*`macroName`* `[`*`count`*`]}...`

Specifies the name of the hard macros that should be included in the blackblob. The `macroName` argument specifies the name of the macro, and `count` specifies the number of times the macro is referenced. Thus, by specifying a value for count, you avoid repeating the macro names for macros that are referenced multiple times.

Default: The default value of *`count`* is `1`.

`-placePorosity` *`percentValue`*

Specifies the percentage of cell area that will be added to the blackblob area during placement. This area allows IPO cells to be placed over the blob area. A placePorosity of 20 implies that 20% of the cell area gets added to the blackblob area during placement for cell padding.

Note: Specifying the placePorosity increases the gateArea, however, the gateArea cannot be greater than the total area.

Default: The default value is `0%`.

`-routePorosity {`*`layerName percentValue`*`}...`

Specifies the routing resources, per layer, that will be reserved for the blackblob. A higher routing porosity implies more routing resources.

Default: The default value is 100%.

`-targetUtil` *`percentValue`*

Specifies the target utilization for the blackblob. This parameter can be used to reduce congestion.

If the -targetUtil is not specified then the gate area is equal to the total blob area. If the targetUtil value is specified then the gate area is calculated based on the total area of the blackblob and the user-specified targetUtil value.

For example, the following command specifies the total area and the target utilization percentage of the blackblob cell `tdsp_core.`

```
specifyBlackBlob -cell tdsp_core -area 110 -targetUtil 60
```

The calculated gate area = (total area * targetUtil) / 100
= (110 * 60) / 100 = 66

Default: The default value is 100%.

Examples

- The following command specifies a blackblob `BlobCell3` with area 140 square microns, a target utilization vale of 70%, and specifies that the macro `blockA2D` should be included in blackblob.

  ```
  specifyBlackBlob -cell BlobCell3 -area 140 -targetUtil 70 -macroList blockA2D 1
  ```

- The following command is similar to the command in the previous example, but specifies that *four* instances of the macro `blockA2D` and *two* instances of the macro `blockB2d` should be included in blackblob.

  ```
  specifyBlackBlob -cell BlobCell3 -area 140 -targetUtil 70
  -macroList blockA2D 4 blockB2D 2
  ```

- The following command specifies a blackblob `BlobCell3` with a gate count of 100, area per gate of 6 square microns, and a target utilization value of 75%.

  ```
  specifyBlackBlob -cell BlobCell3 -gateCount 100 -areaPerGate 6 -targetUtil 75
  ```

- The following commands represent the flexibility with which you can define a blackblob. All of these have equivalent values for total area, gate area, and target utilization.

  ```
  specifyBlackBlob -cell BlobCell3 -area 125 -targetUtil 80
  specifyBlackBlob -cell BlobCell3 -gateArea 100 -targetUtil 80
  specifyBlackBlob -cell BlobCell3 -gateCount 20 -areaPreGate 5 -targetUtil 80
  ```

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Using Blackblobs"

specifyNetWeight

```
specifyNetWeight
     netName ...
     netWeightValue
```

Specifies the priority weighting of a net.

Note: Partition pin assignment honors `specifyNetWeight`. Pins that connect to a net that has a higher net weight are assigned before the pins that connect to a net with a lower net weight.

You can use this command after importing the design.

Parameters

..	Specifies the name(s) of an individual net. You can use wildcards (`*?`) with this parameter.
netWeightValue	Specifies the weighting priority of the net name, with values `512` (highest) through `1` (lowest), or `0` (don't care). *Default:* If you do not specify this parameter, the weight for nets is `2`. Cadence recommends using a value between `1` to `10`. The maximum net weight is `512`. If you set the value greater than `512`, the value will be set at `512`.

Examples

The following commands set the priority net weighting for nets `STRW` to first, `WE2N` to second, and `IOWIN` to third:

```
specifyNetWeight WE2N 9
specifyNetWeight STRW 10
specifyNetWeight IOWIN 8
```

stretchRows

```
stretchRows
     {-left | -right | -left -right}
     [-toCoreEdge]
```

Stretches selected rows.

You can use this command after importing a design.

Parameters

`-left`	Specifies that the left edge of all selected rows should be aligned to the left-most edge among all selected rows.
`-right`	Specifies that the right edge of all selected rows should be aligned to the right-most edge among all selected tows
`-toCoreEdge`	Specifies that row edges should be aligned to the core boundary.

Example

The following command specifies that the left edge of all selected rows should be aligned to the left-most edge among all selected rows.

```
stretchRows -left
```

swapPins

`swapPins`

Switches the locations of two pins.

You can use this command after selecting two pins in the same module in the design display area.

Parameters

None

Related Topics

- "Floorplanning the Design" chapter of the *Encounter User Guide*
  - "Editing Pins"

unfixAllIos

```
unfixAllIos [-incAreaIo]
```

Changes the status of all I/Os from *fixed* to *placed*. If any I/O has a status other than *fixed* prior to running this command, that status will be retained.

You can use this command after importing the design.

Parameters

`-incAreaIo`	Changes the status of all `CLASS PAD AREAIO` cells from a `FIXED` state to a `PLACED` state.

Examples

- The following command changes the status of all I/O pins and I/O cells from a `FIXED` state to a `PLACED` state:

  ```
  unfixAllIos
  ```

- The following command changes the status of all I/O pins, I/O cells, and all `CLASS PAD AREAIO` cells from a `FIXED` state to a `PLACED` state:

  ```
  unfixAllIos -incAreaIo
  ```

Related Topics

- "Data Preparation" chapter of the *Encounter User Guide*
  - "Performing Area I/O Placement"

unplaceAllBlocks

`unplaceAllBlocks`

Unplaces all blocks in the floorplan.

You can use this command after importing the design.

Parameters

None

unplaceAllGuides

`unplaceAllGuides`

Clears all constraints (guides, regions, and fences) from the floorplan.

You can use this command after importing the design.

Parameters

None

unplaceAllInsts

```
unplaceAllInsts
```

Unplaces all instances from the floorplan.

You can use this command after importing the design.

Parameters

None

unplaceGuide

```
unplaceGuide
     {hInstName | groupName}
```

Unplaces hierarchical instances and instance groups from the floorplan. Guides are hierarchical instances in the design and are guided in the floorplan.

You can use this command after importing the design.

Parameters

groupName	Specifies the name of the created group.
hInstName	Specifies the name of the hierarchical instance to be guided.

Examples

- The following command unplaces the guide `MEM_DSCAN` from the floorplan:
  ```
  unplaceGuide MEM_DSCAN
  ```
- The following command unplaces the instance group `adder1` from the floorplan:
  ```
  unplaceGuide adder1
  ```

unplaceGuideConstraints

`unplaceGuideConstraints`

Clears only the guide constraints from the floorplan.

You can use this command after importing the design.

Parameters

None

unsetFixedBlockSize

```
unsetFixedBlockSize
     {* | listOfInstances}
```

Specifies that during floorplan resizing, the size of the specified modules and/or blackboxes can change. Use this command to unset the fixed block size specified with the `setFixedBlockSize` command.

You would use this command after setting the block size of modules and/or black boxes to fixed with the `setFixedBlockSize` command.

Parameters

*	Specifies that the size of *all* the modules and/or blackboxes can change during floorplan resizing.
`listOfInstances`	Specifies that the size of the *specified* modules and/or blackboxes can change during floorplan resizing. Provide a space-separated list of instances.

Example

The following command specifies that the size of `instanceA` and `instanceB` can change during floorplan resizing.

```
unsetFixedBlockSize instanceA instanceB
```

unspecifyBlackBlob

`unspecifyBlackBlob {-blob` *`blobNameList`* `| -all}`

Unspecifies one or more existing blackblobs.

You can use this command after specifying one or more blackblobs.

Parameters

`-all`	Specifies that all the existing blackblobs should be unspecified.
`-blob` *`blobNameList`*	Specifies the name of the blackblob to unspecify. You can specify multiple blackblobs by separating their names with a space.

Example

The following command unspecifies the blackblobs named `tdsp_blockA1` and `tdsp_blockA2`.

```
unspecifyBlackBlob -blob tdsp_blockA1 tdsp_blockA2
```

8

General Commands

bindKey

`bindKey` *`key cmd`*

Enables you to create keyboard shortcuts.

You can use this command at any time after starting the Encounter software.

Parameters

`key`	Specifies the key to use as the shortcut.
`cmd`	Specifies the command to be executed when key is pressed.

Example

Runs the `fit` command when you press `f` on your keyboard:

```
bindKey f "fit"
```

cdump2lef

```
cdump2lef
     -tech techFile
     [-output outFile]
     [-techonly]
     [-log logFile]
     cdumpFile
```

Generates a LEF abstract file from a cdump file.

The *techFile* and *cdumpFile* parameters are mandatory. If the -techonly option is used, then only *techFile* is mandatory.

Note: You must issue this command from a UNIX terminal window. You cannot issue this command from the Encounter console.

You can use this command before you import the design.

Parameters

cdumpFile	Name of cdump file.
-log *logFile*	Changes log file name. *Default*: cdump2lef.log
-output *outFile*	Specifies name (without extension) of output LEF file. *Default*: *base_name_of_cdumpFile*.lef
-tech *techFile*	Specifies the Encounter input technology file.
-techonly	Generates only the LEF process technology file.

Examples

- Basic cdump:

  ```
  cdump2lef -tech enc.tech design.cdump
  ```

- Changing names (from default) of log file and LEF file:

  ```
  cdump2lef -tech e.tech -log ./translation.log -output lefdes.lef design.cdump
  ```

- Generates only LEF technology file:

  ```
  cdump2lef -tech enc.tech -techonly
  ```

changeInstName

```
changeInstName
     -inst instName
     -newBaseName baseName
```

Changes the base name of the specified instance to the given base name.

Note: The `changeInstName` command does not change the path of hierarchy.

Parameters

`-inst` *instName* Specifies the name of the instance.

`-newBaseName` *baseName*

Specifies the new base name to use for the instance.

checkNetlist

`checkNetlist -outfile` *`filename`* `[-includeSubModule]`

Performs several checks on the design netlist and creates a report with information about the following:

- Design Summary
- Design Statistics
- I/O Pad Summary
- Design Rule Checking
- Submodule Port Definition Checking (optional)

You can use this command after the design netlist is imported.

Parameters

`-outfile` *`filename`*	Specifies the name of the file where the report will be written. If no name is specified, the default filename is `checknetlist.rpt`.
`-includeSubModule`	If this option is included, `checkNetlist` checks the port definitions for any submodules.

Example

Error conditions are indicated by an asterisk (*) preceding the reported statistic, such as

```
    Number of Ports Connect to Core Cells        *: 57
    Number of Ports Connect to multiple Pads     *: 0
    Number of Floating Ports                     *: 0
```

Warnings, or possible errors are indicated by a question mark in front of the statistic, such as

```
    Number of undefined Floating Pins             ?: 0
```

The following example shows the information contained in the generated by `checkNetlist` report.

```
Design: TOPModule

------ Design Summary:
```

```
Total Standard Cell Number   (cells) : 5908
Total Block Cell Number      (cells) : 4
Total I/O Pad Cell Number    (cells) : 0
Total Standard Cell Area     ( um^2) : 133864.32
Total Block Cell Area        ( um^2) : 366346.56
Total I/O Pad Cell Area      ( um^2) : 0.00

------ Design Statistics:

Number of Instatnces           : 5912
Number of Nets                 : 6217
Average number of Pins per Net : 3.69
Maximum number of Pins in Net  : 541

------ I/O Pad summary

Number of Primary I/O Ports    : 57
Number of Input Ports          : 28
Number of Output Ports         : 29
Number of Bidirectional Ports  : 0
Number of Power/Ground Ports   : 0
Number of Ports Connect to Core Cells        *: 57
Number of Ports Connect to multiple Pads     *: 0
Number of Floating Ports                     *: 0

------ Design Rule Checking:

Number of Output Pins connect to Power/Ground *: 0
Number of Input Floating Pins                 *: 1
Number of Output Floating Pins                 : 245
Number of InOut Floating Pins                  : 0
Number of UniPort Floating Pins                : 0
Number of undefined Floating Pins            ?: 0

Number of Multiple Driving Nets               *: 1
Number of Parallel Driving Nets                : 1
Number of Tristate Driving Nets                : 0
Number of Input Floating Nets(No FanIn)       *: 0
Number of Output Floating Nets(No FanOut)      : 1
Number of Hign Fanout Nets (>50)               : 3
```

```
------ Sub Module Port Definition Checking

* module arb has 3 port define error
* module accum_stat has 1 port define error
* module tdsp_core_glue has 128 port define error
```

createSnapshot

```
createSnapshot
     [-help]
     -dir string
     -name string
     [-overwrite]
```

With this command you can create snapshots of the design for all three views (FloorPlan, Amoeba, and Placement) at any stage of the design flow. The command creates GIF images, for all the design views, on the same scale as can be seen in the Encounter software. It can also save QoR parameters like WNS ,TNS, Placement Overlap, and Congestion at any stage of the design flow. Once the snapshot is created, you can view it using viewSnapshot.

Parameters

`-dir string`	Specifies the directory in which you want to save the snapshot.
`-help`	Outputs a brief description that includes type and default information for each `createSnapshot` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man createSnapshot`.
`-name string`	Specifies the name of snapshot and the files will be created, like `ss_string_view.gif` for all three views.
`-overwrite`	Overwrites the existing snapshot with new one.

Note: This command will not work in the `-nowin` mode.

Examples

- The following command will create a design snapshot in the directory `/work/user1`. Before creating the snapshot, it will create the following files:
  - `ss_cts.fplan.gif` for fplan view
  - `ss_cts.amoeba.gif` for amoeba view
  - `ss_cts.place.gif` for place view

  ```
  createSnapshot -dir /work/user1 -name cts
  ```
- The following command will create a snapshot under the name `cts`. If there exists a snapshot under this name in the directory `/work/user1`, it will overwrite it.

```
createSnapshot -dir /work/user1 -name cts -overwrite
```

Related Topics

For more information, see the "Tools Menu" chapter in the *Encounter Menu Reference*.

- Create Snapshot
- View Snapshot

dehighlight

```
dehighlight
     [-index indexValue | -selected]
```

Dehighlights all the highlighted objects. You can also selectively dehighlight selected objects or objects with specified index value.

Parameters

`-index indexValue`	Specifies that the objects with the index value provided will be dehighlighted. The index value is the value of the Highlight Set as specified through the Edit – Highlight Selected menu or through the highlight command. The Edit – Highlight Selected menu provides eight highlight patterns to choose from. The value of index should, therefore, be a number be from 1 to 8. As an example, if you specify a value of 3 for index, all objects that are highlighted as Highlighted Set 3 will be dehilighted.
`-selected`	Specifies that all selected objects will be dehighlighted.

Related Topics

- Edit Menu chapter in the *Encounter Menu Reference*
  - Highlight Selected
  - Clear Highlight
  - Edit Highlight Color

Examples

- The following command dehighlights all highlighted objects.

  ```
  dehighlight
  ```

- The following command dehighlights all highlighted objects that have been highlighted as Highlight Set 4 through the Edit – Highlight Selected menu or through the highlight command.

  ```
  dehighlight -index 4
  ```

- The following command dehighlights all highlighted objects that are currently selected in the display window.

```
dehighlight -selected
```

deleteDanglingPort

`deleteDanglingPort`

Deletes ports that are disconnected from both the inside and outside of a module, and ports that are connected within the module but disconnected outside the module.

Parameters

None.

encMessage

```
encMessage {warning | info | debug} {0 | 1}
```

Specifies whether to enable the display of warning, information, or debug messages.

Parameters

`warning | info | bug`	Specifies the type of message you want to enable or disable.
`0 | 1`	Disables (`0`) or enables (`1`) the display of the messages.

Example

- The following command disables the display of all information messages:

  ```
  encMessage info 0
  ```

findNetsInBox

`findNetsInBox` *`llx lly urx ury`*

Identifies the nets that have wires intersecting within the specified sector of the bounding box.

You can use this command in the CCAR interface to locate all nets in a given bounding box.

Parameters

`llx`	Specifies the lower left x coordinate of the bounding box.
`lly`	Specifies the lower left y coordinate of the bounding box.
`urx`	Specifies the upper right x coordinate of the bounding box.
`ury`	Specifies the upper right y coordinate of the bounding box.

Example

The following command specifies that the software should look for intersecting wires in the area specified (50, 50, 1000, 1000):

```
findNetsInBox 50 50 1000 1000
```

getBuildArch

`getBuildArch`

Reports whether the Encounter session is running in 32-bit or 64-bit mode.

Parameters

None

getCheckMode

```
getCheckMode
     [-all]
     [-extraction]
     [-floorplan]
     [-globalNet]
     [-integrity]
     [-io]
     [-library]
     [-mgrid]
     [-netlist]
     [-placement]
     [-quiet]
     [-route]
     [-sroute]
     [-tapeOut]
     [-timingGraph]
     [-vcellnetlist]
```

Displays the following information about a specified `setCheckMode` parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setCheckMode` parameters.

Parameters

`-parameter_names`

Displays information for the specified parameters. You can specify one or more parameters.

See `setCheckMode` for descriptions of the data checking parameters you can specify.

`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setCheckMode` parameters in Tcl list format.

Examples

- The following command displays the current setting for the `-extraction` parameter:

  ```
  getCheckMode -extraction
  ```

  The software displays the following information:

  ```
  -extraction false    # bool, default=false
  false
  ```

- The following command displays the current setting of the `-extraction` parameter in Tcl list format:

  ```
  getCheckMode -extraction -quiet
  ```

  The software displays the following information:

  ```
  false
  ```

- The following command displays the current settings of the `-floorplan` and `-placement` parameters:

  ```
  getCheckMode -floorplan -placement
  ```

  The software displays the following information:

  ```
  -floorplan false # bool, default=false
  -placement false # bool, default=false
  {floorplan false} {placement false}
  ```

- The following command displays the current settings of the `-floorplan` and `-placement` parameters in Tcl list format:

  ```
  getCheckMode -floorplan -placement -quiet
  ```

  The software displays the following information:

  ```
  {floorplan false} {placement false}
  ```

getCompressLevel

`getCompressLevel`

Displays the current compression speed level used for compressing large data files. Compression speed is controlled with the `setCompressLevel` command.

getLayerPreference

```
getLayerPreference
     layer_name
     {-isVisible | -isSelectable | -color | -lineWidth | -stipple | -stippleData}
```

Provides information about a single preference for a single object.

See setLayerPreference on page 666 for descriptions of how to set layer preferences.

Parameters

`-color`	Provides information about the color of the object. A color name or a numeric RGB value is returned.
`-isSelectable`	Provides information about whether the object is selectable. If a value of `1` is returned, the object is selectable. If a value of `0` is returned, the object is not selectable.
`-isVisible`	Provides information about whether the object is visible. If a value of `1` is returned, the object is visible. If a value of `0` is returned, the object is not visible.
`layer_name`	Provides the name of the layer or the object type.
`-lineWidth`	Provides information about the number of pixels in the border of the specified object. An integer value is returned.
`-stipple`	Provides information about the stipple pattern of the object. A stipple name is returned.
`-stippleData`	Provides information about the width and height of the stipple pattern. This information is in x-window bitmap format.

Example

Returns the color for `M1` objects:

```
getLayerPreference M1 -color
```

getVersion

`getVersion`

Returns the Encounter software version number.

Parameters

None

help

`help {commandName | -k keyword | string}`

Displays Encounter commands and their parameters.

After typing `help` and the command name, and then pressing the `Return` key, the command syntax output is written in the log file and displayed in the Encounter console.

You can also use the `help` command for keyword searches and partial string searches, without concern for case sensitivity.

Parameters

commandName	Displays help for the specified command name. If you do not specify a command name, all officially supported Encounter commands are displayed.
-k *keyword*	Searches for the specified keyword and displays all commands associated with that keyword.
string	Displays all commands that contain the specified series of characters.

Examples

- The following command prints the command syntax for the `trialRoute` command:

  ```
  help trialRoute
  ```

- The following command displays all commands that include “place” as a keyword:

  ```
  help -k place
  ```

- The following command displays all commands that contain the string “bud” (such as `deriveTimingBudget`):

  ```
  help bud
  ```

highlight

```
highlight
     -index indexValue
     [-color colorValue]
     [-pattern patternValue]
```

Highlights selected objects with the Highlight Set available through the Edit – Highlight menu. The value for the index corresponds to the Highlight Set that will be applied. For example, a value of 4 for the index specifies that all selected objects will be highlighted with the same color and pattern as available for the Highlight Set 4 in the Edit – Highlight Selected menu.

You can also provide a specific color and/or pattern to be used. All selected objects will be highlighted with the color and/or pattern that you specify.

Note: If you specify a color and or pattern, the Highlight Set corresponding to the index value will be updated to use the color and pattern specified. For example, if you specify an index value of 5, color `green`, and pattern `backslash`, the Highlight Set 5 will get updated to have the color `green` and the pattern `backslash`.

You can dehighlight the highlighted objects through the createSnapshot command or through the Edit – Dehighlight menu.

Parameters

`-index` *`indexValue`*	Specifies the index value for the specified object. All objects that are selected will be highlighted with the same color and pattern as defined in corresponding Highlight Set in the Edit – Highlight Selected menu As an example, if you specify a value of 3 for index, all objects that are selected will be highlighted with the same color and pattern as available for the Highlight Set 3 in the Edit – Highlight Selected menu. The Edit – Highlight Selected menu provides eight highlight patterns to choose from. The value of index should, therefore, be a number be from 1 to 8.

`-color colorValue` Specifies the color value for the specified object. The color value is the same as that available through the Edit – Highlight Color form.

You can also use the Red Green Blue (RGB) value in the following format: #*V1V2V3* where V1, V2, and V3 are the hexadecimal values for the red, green, and blue color components respectively. For example, you can specify the value #CC66FF.

Default: The color defined for the Highlight Set corresponding to the index value is used.

`-pattern patternValue`

Specifies the pattern value for the specified object. The pattern value is the same as the Stipple Value available through the Edit – Highlight Color form.

Default: The pattern (Stipple Value) for the Highlight Set corresponding to the index value is used.

Related Topics

- Edit Menu chapter in the *Encounter Menu Reference*
  - Highlight Selected
  - Clear Highlight
  - Edit Highlight Color

Examples

- The following command highlights all selected objects with the same color and pattern as defined for the Highlight Set 4 in the Edit – Highlighted Selected Menu.

```
highlight -index 4
```

- The following command highlights all selected objects with the color `green` and the pattern `dot4`. The index values is set to 5 and, therefore, the command will also change the definition of the Highlight Set 5 (Edit – Highlight Selected menu) to have color `green` and pattern `dot4`.

```
highlight -index 5 -color green -pattern dot4
```

- The following command highlights all selected objects with the color corresponding to the RGB hexadecimal value `#33FF66` and the pattern `horizontal`. The index values is

set to 3 and, therefore, the command will also change the definition of the Highlight Set 5 (Edit – Highlight Selected menu) to have color #33FF66 and pattern `horizontal`.

```
highlight -index 3 -color #33FF66 -pattern horizontal
```

pauseScript

`pauseScript`

Pauses the execution of the commands in a script.

When running a set of commands through a script, you can use this command to pause the execution of the commands. Specify this command in the script at the point where you want to pause the execution. You can resume the execution of the commands in the script by pressing the `ENTER` key.

The `pauseScript` command should be used inside a script, and runs when it is executed in the script.

Parameters

None

placeCursor

`placeCursor` *`x y`*

Places the cursor at the specified location in the main window.

You can specify this command at any time after starting the encounter software.

Parameters

x	Specifies the x coordinate of the cursor location.
y	Specifies the y coordinate of the cursor location.

Example

The following command places the cursor at the coordinates (133, 277).

```
placeCursor 133 277
```

readlef

```
readlef
     [-m map_file_name]
     [-keepPinName]
     [-allWarning]
     [-template]
     [-c layer_number]
     [-cutVia]
     [-list list_file_name]
     [-output output_file_name]
     lefFile ...
```

(UNIX prompt command)

Translates LEF format file(s) into Cdump format files and a technology file for Encounter.

You can use this UNIX prompt command any time.

Parameters

`-allWarning`	This option displays all warning messages.
`-c` *`layer_number`*	Specifies the number of the metal layer covered by a route obstruction. The `readlef` command creates a whole layer blockage for the specified layer and then does the pin cut. For obstructions on multiple layers, you can specify this option multiple times (`-c 5 -c 6`, for example).
`-cutVia`	Calculates the pin cut spacing based on the via enclosure when the `readlef` command does the pin cut.
`-keepPinName`	Preserves the pin names and does not do any kind of conversion.
`lefFile`	One or more files that contain the LEF data. If you specify multiple files, the first one must contain the technology data.
`-list` *`list_file_name`*	Specifies the file with the list of the names of all input files. This file contains one file name per line.

`-m` *`map_file_name`*
: Specifies a file that maps the cell to the cellclass. For LEF files in which `cellclass` is not specified, this parameter provides the side file to map each cell to the correct cellclass.

  This file is optional and used only when Classes are either missing in the LEF file or need remapping.

`-output` *`output_file_name`*
: Specifies the base name for the files that contain the results.

`-template`
: Prints a sample cellMapFile for the purpose of mapping cells to separate Cdump files for standard cells, blocks, and I/Os.

Examples

- Creates three cdump files and a technology file from a single LEF format file:

  ```
  readlef myLefFile
  ```

  The files created are:

  - `myLefFile.cdump`

    Contains standard cells

  - `myLefFile.block.cdump`

    Contains block cells

  - `myLefFile.io.cdump`

    Contains I/O pad cells

  - `myLefFile.tech`

    Technology file

- Creates three cdump files and a technology file from LEF files which has the cell's header file first followed by the cell data files:

  ```
  readlef lef.header cell1 cell2 cell3 cell4 ...
  ```

redo

```
redo
```

Returns the design to the state it was in immediately prior to issuing the `undo` command. It can only be used to reinstate power planning, wire editing, and selected floorplanning functions that were removed with the `undo` command.

You can use this command after you use the `undo` command for any of the following text commands:

- `addRing`
- `addStripe`
- `alignInst`
- `connectRingPin`
- `editDelete`
- `editPowerVias`
- `flipInst`
- `repairStripes`
- `rotateInst`
- `shiftInst`
- `snapFPlanIO`
- `snapFPlan`
- `spaceInst`
- `undo`

Parameters

None

reportDanglingPort

```
reportDanglingPort
     -outfile fileName
```

Reports ports that are disconnected from both the inside and outside of a module, and ports that are connected within the module but disconnected outside the module.

Parameters

`-outfile`	Specifies the name of the report generated by this command.

Examples

The following command writes information on dangling reports to file `myreport`:

```
reportDanglingPort -outfile myreport
```

reportFanin

```
reportFanin
     [-outfile file]
     {portNames | pinNames | netNames}
```

Reports fanin cone of the specified design objects in the current module. The generated report contains all information related to the fanin cone, for example, from pin, to pin, and level in the module.

Parameters

`-outfile file`	Specifies the name of the output file.
`portNames | pinNames | netNames`	Specifies the Tcl list of pin, port or net names.

reportFanout

```
reportFanout
     [-outfile file]
     {portNames | pinNames | netNames}
```

Reports fanout cone of the specified design objects in the current module. The generated report contains all information related to the fanout cone, for example, from pin, to pin, and level in the module.

Parameters

`-outfile file`	Specifies the name of the output file.
`portNames | pinNames | netNames`	Specifies the Tcl list of pin, port or net names.

reportGateCount

```
reportGateCount
     [-module moduleName]
     [-level level]
     [-limit gateCount]
     [-stdCellOnly]
     [-outfile fileName]
```

Reports the size of the imported design, measured in gate counts.

The software calculates the gate count for a module using the following equation:

gateCount = moduleArea / gateSize

Where:

- moduleArea is the area of the module. By default, this is the sum of the areas of all instances inside of the module, including standard cells, blocks, and I/O cells. If you specify the `-stdCellOnly` parameter, moduleArea is the sum of the areas of the standard cells inside the module only.
- gateSize is calculated as follows:

  gateSize = dbgStdCellHgt x dbgDBUPerIGU x dbgSitesPerGate

  Where:

  - dbgStdCellHgt is the standard cell row height.
  - dbgDBUPerIGU is the *M2* layer pitch.
  - dbgSitesPerGate is a user-defined global variable that determines the gate size the software assumes when calculating the gate count. For example, the default value of 3 means the assumed the gate size is equal to 1 standard cell row height and 3 *M2* layer pitch widths. You can change this value to adjust the assumed gate size for each specific design.

You can use the `reportGateCount` command after importing the design.

Parameters

`-level` *`level`*	Expands the module to the specified level. For example, if you specify `1`, CTS expands the specified module by one level. The gate counts of all “children” modules of the module that satisfy the `-limit` parameter are included in the gate count report. If you specify `2`, CTS expands the module by two levels. The gate counts of all “children” and “grandchildren” modules of the module are included in the gate count report. *Default*: Module is not expanded, and CTS reports only the gate count of the module itself. (This is equivalent to specifying a level of `0`.)
`-limit` *`gateCount`*	Instructs CTS to ignore modules with gate counts below the specified limit. Such modules are ignored during expansion and reporting. *Default*: CTS ignores modules with 1000 or fewer gates during expansion and reporting.
`-module` *`moduleName`*	Specifies the module for which to generate a gate count report. *Default*: Generates a gate count report for the hierarchical instance corresponding to the top cell of the design.
`-outfile` *`fileName`*	Writes out the gate count report to the specified file name. The default file extension is `.gateCount`. *Default*: Displays the gate count report *only* in the Encounter console.
`-stdCellOnly`	Calculates gate count based only on standard cells within the specified module. Blocks and I/O cells within the module are ignored.

reportNetLen

```
reportNetLen
```

Used to report half the total net bounding box perimeter length for the design.

You can use this command after placing the design.

Parameters

None

reportNetStat

```
reportNetStat
```

Used to report the design statistics, such as number of cells, nets, pins, and I/Os.

You can use this command after importing the design.

Parameters

None

saveTechFile

```
saveTechFile fileName
```

Saves the current technology information to a file. This is used to dump the technology information after design import and all the key control words/strings. All the control words/strings are listed in the header section at the top of the dumped file.

You can use this command after importing the design.

Parameters

fileName	The name of dumped technology file.

Example

Writes the current technology information to a file:

saveTestcase

```
saveTestcase
     -name testcaseName
     -dir directoryName
     -rc
     -merge
     -overwrite
     -gzip
```

Saves your design as a standalone testcase that includes both the design and the library files. All database references to files in the generated testcase are self-contained, such that the database can be loaded into Encounter on a different network.

You can use this command after loading a design.

Note: Exit Encounter after using the command. This is to avoid a possibility where `saveDesign` may create a configuration file with a timing constraints file pointing to the testcase directory. Given that the testcase directory is considered temporary, and may be gzipped or even deleted, a pointer to its data is not desirable. Exiting Encounter and re-loading a previously saved database will avoid this possibility.

Parameters

`-name` *`testcaseName`*

Specifies the name of the testcase to be created. The path to the saved testcase location is:

`directoryName/testcaseName`

If a previously generated testcase exists at the specified location, the older testcase will be renamed before the new testcase is saved.

Default: testcase

`-dir` *`directoryName`*

Specifies the directory in which to save the testcase. Both, absolute and relative directory paths, may be used.

Default: Current working directory

`-rc`	Specifies that RC extraction data should be saved. *Default*: RC parasitics files are not saved. **Note:** This option works with these limitations: ■ The `-rc` parameter does not have save RC extraction data if the design is not extracted or if you extract the data using the default extraction engine. ■ The previously saved RC data can only be restored on systems that share the same platform and architecture. For example, if you saved the RC data on a Linux platform you can not restore it on a IBM AIX platform. Similarly, RC data saved on a 64-bit system cannot be restored on a 32-bit system.
`-merge`	Specifies that a single script should be generated containing both shell executable and Encounter TCL commands. *Default*: Two script files are generated. A shell executable file and an Encounter TCL script.
`-overwrite`	Specifies that existing files should be overwritten. *Default*: Existing testcase files are renamed before new testcase is saved.
`-gzip`	Specifies that the generated self contained design should be stored in a gzip compressed tar file. The generated test case directory and files are deleted after gzip compression is complete. *Default*: If you do not specify this parameter, the generated test case is not gzipped.

Example

- The following command creates a self contained design in the current working directory under the testcase directory.

  ```
  saveTestcase
  ```

- The following command creates a gzip compressed tar file `/tmp/testcase.tar.gz` containing a self contained testcase:

  ```
  saveTestcase -dir /tmp –gzip
  ```

selectObjByProp

```
selectObjByProp
     objectType
     expression
```

Logs the select operations performed through the Find/Select Object form (*Edit — Find/Select Object*).

Parameters

objectType	The name of the object(s) on which the select operation is performed. The object types are: ■ Instance ■ Pin ■ Net ■ Module ■ InstanceGroup ■ Bump
expression	The selection criteria specified through the Find/Select Object form.

Examples

- The following command logs the select operation for all instances of type BLOCK in the Edit – Find/Select Objects form. In this case, the value of the Mode field was set to *Single* in the Find/Select Objects form.

  ```
  selectObjByProp Instance <Type>=<Block>
  ```

- The following command logs the select operation for all I/O pins that are on the layer M2 and whose status is not *placed*. In this case, the value of the Mode field was set to *And* in the Find/Select Objects form.

  ```
  selectObjByProp Pin {And(<Type>=<I/O Pin>,!<Status>=<Placed><layer>=<M2>)}
  ```

- The following command logs the select operation for all modules whose constraint type is Fence or Region. In this case, the value of the Mode field was set to *OR* in the Find/Select Objects form.

```
selectObjByProp Module {OR(<Constraint Type>=<Fence>,<Constraint
Type>=<Region>)}
```

- The following command logs the select operation for all pins of the modules DTMF_INST/TDSP_CORE_INST and DTMF_INST/ARB_INST that are on the top side and have a status of *placed*, *fixed*, or *cover.* In this case, the value of the Mode field was set to *Multiple* in the Find/Select Objects form.

```
selectObjByProp Pin
And(And(!<Status>=<Unplaced>,<Side=<Top>),OR(<Module>Match<DTMF_INST/
TDSP_CORE_INST>,
<Module>Match<DTMF_INST/ARB_INST>))
```

setCheckMode

```
setCheckMode
     [-help]
     [-reset]
     [-all {true | false}]
     [-checkIlm {true | false}]
     [-extraction {true | false}]
     [-floorplan {true | false}]
     [-globalNet {true | false}]
     [-integrity {true | false}]
     [-io {true | false}]
     [-library {true | false}]
     [-mgrid {true | false}]
     [-netlist {true | false}]
     [-placement {true | false}]
     [-route {true | false}]
     [-sroute {true | false}]
     [-tapeOut {true | false}]
     [-timingGraph {true | false}]
     [-vcellnetlist {true | false}]
```

Sets the data checks that the software performs.

If the software finds missing or incorrect data during the check, it

- Does not proceed with the functionality that would be affected by the incomplete or incorrect data
- Generates an error message in the Encounter console

Use the `getCheckMode` command to display the current settings for the `setCheckMode` command.

Use the `setCheckMode` command immediately before importing the design, or before performing a functionality.

Parameters

`-all {true | false}` Performs all data checks.

Default: `false`

`-checkIlm {true | false}`

Checks the ILM model.

Default: `false`

`-extraction {true | false}`

Checks that all data required for extraction are correct.

Note: Although this parameter instructs the Encounter software to check that routing is complete, this parameter does *not* check wiring for DRC violations.

Default: `false`

`-floorplan {true | false}`

Checks the floorplan.

Default: `false`

`-globalNet {true | false}`

Checks instance pin and power/ground pin net connections.

Default: `false`

`-help` Outputs a brief description that includes type and default information for each `setCheckMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setCheckMode`.

`-integrity {true | false}`

Note: No information available.

Default: `false`

`-io {true | false}` Checks the I/O pads.

Default: `false`

`-library {true | false}`

Checks the physical (LEF) and timing libraries.

Default: `false`

`-mgrid {true | false}`

Checks that geometries are on the manufacturing grid.

Note: The software does not stop this check, even if there are problems in the DEF file. The software generates an error message if it finds any problems in the DEF file.

Default: `false`

`-netlist {true | false}`

Checks the netlist.

Default: `false`

`-placement {true | false}`

Checks the placement of the design.

Default: `false`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setCheckMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-route {true | false}`

Checks whether power and ground pins are connected to power and ground nets before power routing. The software also checks whether all tie high and tie low pins are connected to global power and ground nets before detailed routing.

Default: `false`

`-sroute {true | false}`

Checks special routes.

Default: `false`

`-tapeOut {true | false}`

Runs the software in tapeout mode instead of in protyping mode.

In tapeout mode, the software stops running if it detects errors that would seriously affect the overall integrity of results. In prototyping mode, the software does not stop if it detects such errors.

Default: `false`

`-timingGraph {true | false}`

Checks library cells for timing information.

Default: `false`

`-vcellnetlist {true | false}`

Checks for tie-high and tie-low output pins when reading the Verilog netlist.

The software stops the flow if tie-high or tie-low output pins are found because connecting an output signal pin to a power or ground creates a potential short circuit. If all of the tie-high or tie-low output pins are actually the driver pins for power or ground nets, specify `-vcellnetlist false`, so that the flow can continue.

Note: To turn off the checking, you *must* set this parameter before loading the configuration file.

Default: `false`

Examples

- The following command turns on all data-checking modes except the pre-power-routing data checking:

  ```
  setCheckMode -all true -route false
  ```

- The following command checks a DEF file for objects that are off the manufacturing grid.

  ```
  setCheckMode -mgrid true
  ```

  The software generates error messages such as the following:

  ```
  **ERROR: Line 56: OffMGrid: Track start value (566) is not on Manufacturing
  Grid.
  **ERROR: Line 446: OffMGrid: Via (-10000, -10000) (10006, 10006) named
  via3Array_1 is not on Manufacturing Grid.
  ```

```
**ERROR: Line 902: OffMGrid: Component location (130686, 171366) of inst
iblockN20/iregNS/yi_reg_1 is not on Manufacturing Grid.
```

- The following command resets the `-mgrid` parameter to its default value:

  ```
  setCheckMode -reset -mgrid
  ```

- The following command resets all `setCheckMode` parameters to their default values:

  ```
  setCheckMode -reset
  ```

setCompressLevel

`setCompressLevel` *`level`*

Specifies the compression speed to use when compressing large data files. A faster compression speed creates a less compressed file. A slower compression speed creates a more compressed file. By default, the software uses the `gzip` default speed value.

Parameters

`level`

Specifies the speed level to use when compressing files.

You can specify a value between 0 and 9, where:

`0`	Performs no compression.
`1`	Performs the quickest compression, and least compressed files.
`9`	Performs the slowest compression, and most compressed files.

Note: To return to the `gzip` default value (`Z_DEFAULT_COMPRESSION`), specify `-1`.

setLayerPreference

```
setLayerPreference
     layer_name
     [-isVisible {1 | 0}]
     [-isSelectable {1 | 0}]
     [-color {color_name | color_value}]
     [-lineWidth {lw}]
     [-stipple stipple_name | -stippleData width height "stipple_data"]
```

Sets a new preference for the specified object or layer.

See getLayerPreference on page 637 for descriptions of how to display layer preferences.

Parameters

`-color`	Specifies a color for the object. Specify either a color name or a numeric RGB value.
`-isSelectable`	Specifies whether the object is selectable. Specify `1` to make the object selectable. Specify `0` to make the object non-selectable.
`-isVisible`	Specifies whether the object is visible. Specify `1` to make the object selectable. Specify `0` to make the object invisible.
`layer_name`	Specifies the name of the layer or the object type. The valid layer names are `M#`, where `#` is `0` through `1`, and `M#AVio`, `M#Cont`, `M#ContPin`, `M#ContRB`, `M#Pin`, `M#RB`, `M#RTrk`, and `M#Vio`. The valid object types are `blackBox`, `block`, `blockHalo`, `bump`, `bump1`, `bump2`, `bump3`, `cellBlkgObj`, `channel`, `clkTree`, `congest`, `congestH`, `congestObj`, `congestV`, `datapath`, `fence`, `gcellOvflow`, `guide`, `hinst`, `inst`, `ioSlot`, `ioRow`, `macroSitePattern`, `metalFill`, `mGrid`, `net`, `netRect`, `nonPrefTrackObj`, `obstruct`, `pinObj`, `pinText`, `power`, `powerNet`, `ptnFeed`, `ptnPinBlk`, `pwrdm`, `pwrdm1`, `pwrdm2`, `pwrdm3`, `layerBlk`, `region`, `routeGuide`, `ruler`, `screen`, `select`, `sitePattern`, `stdRow`, `term`, `text`, `trackObj`, `userGrid`, and `violation`.
`-lineWidth`	Specifies the number of pixels in the border of the specified object. Specify this value as integer.

`-stipple`	Specifies information about the stipple pattern of the object. The valid stipple names are `None`, `Horizontal`, `Vertical`, `Grid`, `Slash`, `Backslash`, `Cross`, and `Brick`.
`-stippleData`	Specifies the stipple information for the object. Specify this information in x-window bitmap format.

Example

The following command sets a layer preference for the module object in green with dotted stipple pattern (8 by 4 bitmap with dot at every 4 pixels):

```
setLayerPreference hinst -color green -stippleData 8 4 “0x00 0x11 0x00 0x00”
```

setLicenseCheck

```
setLicenseCheck
     [-help]
     [-optionList {license1 license2 …}]
     [-wait time_in_minutes]
     [-default]
     [-checkout license]
     [-status]
     [-existOnServer]
```

Manages licenses for product options. Specifies licenses to check out, license wait time, and the status of all product option licenses.

Note: This command does not manage multi-CPU licenses. For information on the commands that manage multi-CPU licenses, see "Multiple-CPU Processing Commands".

Parameters

`-checkout` *`license`*	Checks out the licenses for the specified product options when this command runs.
`-default`	Restores the default. values for this command.
`-existOnServer`	Outputs a table that provides the following information for each of the licenses that exist on the license server. ■ License name ■ Product number ■ Product name ■ License string ■ Version
`-help`	Outputs a brief description that includes type and default information for each `setLicenseCheck` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man setLicenseCheck`.

`-optionList {`*`license1 license2 …`*`}`

Specifies licenses to check out dynamically. Dynamic licenses are not checked out until they are needed. If you specify a license that is not allowed with your base license, or a license that is not available, the software does not check out that license and instead issues a warning message.

If you specify more than one license, begin and end the option list with double quotation marks or braces.

`-status`

Outputs the following information:

- The name of the base license that is checked out
- The current wait time
- A list of the allowed license options
- A list of the available license options
- A list of the currently checked-out license options.

`-wait` *`time_in_minutes`*

Specifies the amount of time the system waits for a license to become available. If the license is available in less than the specified wait time, the system checks out the next needed license without waiting.
Default: `0` (no wait time)
Value range: `0` to `10,000`

Examples

- The following command outputs a table that provides information about the licenses that exist on the license server:

  ```
  setLicenseCheck -existOnServer
  ```

  The system outputs the following information:

  ```
  Name  Prod #  Product Name License string             Version
  epsxl EPS200  EPS-XL       Encounter_Power_System_XL  8.1
  epsl  EPS100  EPS-L        Encounter_Power_System_L   8.1
  etsxl FE725   ETS-XL       Encounter_Timing_System_XL 8.1
  etsl  FE625   ETS-L        Encounter_Timing_System_L  8.1
  cndc  CM00500 Celtic_NDC   Celtic_NDC                6.1000
  encng ENG100  ENC-NG       Encounter_NG100            1.0000
  ```

- The following command specifies that licenses for Encounter Timing System-L and Encounter Timing System-XL may be checked out. Encounter Timing System-L licenses are checked out before Encounter Timing System-XL licenses.

  ```
  setLicenseCheck -optionList {etsl etsxl}
  ```

- The following command outputs license status and wait time and lists the optional licenses that can be checked out and the licenses that are checked out:

  ```
  setLicenseCheck -status
  ```

  The system outputs the following information:

  ```
  Current checked out base license : socegxl (SOC_Encounter_GXL 8.1).
  When checking-out an optional license, wait for up to : 0 minutes.
  The base license allows the following optional licenses : epsxl epsl etsxl etsl
  cndc encng
  The automatic optional license check-out is limited by -optionList to : epsxl
  epsl etsxl etsl cndc encng
  Current checked out optional licenses : none
  Summary of all checked out licenses:
  SOC_Encounter_GXL
  ```

- The following command sets the wait time to 5 minutes:

  ```
  setLicenseCheck -wait 5
  ```

Related Topics

- Products and Licensing Information chapter in the *Encounter User Guide*
- Getting Started chapter in the *Encounter User Guide*

setMessageLimit

`setMessageLimit` *`count`*

Limits error and warning messages to a specified number per unique ID. Any occurrences beyond the limit are suppressed. The default limit is 20 per message ID. The default limit is applicable only for `ERROR*` and `WARNING*` messages; there is no default limit for `verify*`, `check*`, and `INFO*` messages.

Parameters

`count`	The maximum number of messages to be displayed for each unique ID. The value of *`count`* should be greater than zero.

Example

The following command sets a limit of 10 messages per unique ID for all error and warning messages:

```
setMessageLimit 10
```

setPreference

`setPreference` *`preference_name value`*

Used to define preferences, such as where commands issued during a session are logged (command, log, or screen).

When writing to log and screen, the command are preceded with `<CMD>`.

Parameters

`preference_name`	The name of the preference you want to set. To see a list of preferences, select *Design – Preferences* from the menu, then click *OK*. The list of preference values appears in the `.cmd` file.
`value`	The value for the preference. For example, the `CmdLogMode` preference has the following values. `0`—write to command only 1—write to command and log (default) 2—write to command, log, and screen

Example

Writes log to command, log, and screen:

```
setPreference CmdLogMode 2
```

setTopCell

`setTopCell` *`topcell_name`*

Switches the design to the partition specified by the *`topcell_name`*.

Note: Cadence recommends that you use the same name for the top cell and the partition.

You can use this command only after you flatten the partition.

Parameters

`topcell_name`	The name of the top cell that indicates a different partition.

setWindowPreference

`setWindowPreference` *`window_name`* `{on | off}`

Enables or disables windows (panels) from the Encounter main display. You can save these window settings in the `.enc` file in your home directory. The settings are automatically loaded when invoking the Encounter main window.

Parameters

`window_name` `{on | off}`

The name of the window you want to turn on or off in the Encounter main display. You can select from the following:

- `properties`—controls the Attribute Viewer panel.
- `status`—controls the message bar display, which shows object identifier or coordinate information.
- `toolbox`—controls the tool widget display.
- `viewColor`—controls the visibility, selectability, and color display of various attributes.
- `world`—controls the satellite window display.

Example

The following command turns on the tool widget display:

```
setWindowPreference toolbox on
```

summaryReport

```
summaryReport
     [ -noText
         [-outdir directoryName]
         [-browser]
     | -noHtml
         [-outfile fileName]
     | [-outdir directoryName]
       [-browser]
     ]
```

Reports statistics for the entire design, or a selected object in the design. A report on the entire design includes statistics for the following categories:

- General design information
- General library information
- Netlist information
- Timing information
- Floorplan/Placement information

To display information about a selected object in the design, you must select the object first before using `summaryReport`. The information contained in the report varies depending on the object you select. You can select a module, an instance, or a net.

You can use this command at any time after importing a design.

Parameters

`-browser`	Opens a browser and displays the HTML version of the report.
`-noHtml`	Generates only a text version of the report. **Note:** If you specify `-noHtml`, you cannot specify `-noText`. *Default:* Generates both an HTML and a text version of the report.

`-noText`

Generates only an HTML version of the report.

Some statistics in the sections on the main HTML report page include links to more detailed HTML report pages. For example, if you click on the *# Layers* link under the *General Library Information* section of the HTML report, the software displays a detailed HTML report on the layers in the design.

For an example of an HTML report, see "HTML Summary Report" in the *Encounter Menu Command Reference*.

Note: If you specify `-noText`, you cannot specify `-noHtml`.

Default: Generates both an HTML and a text version of the report.

`-outDir` *`directoryName`*

When `-noText` is specified, saves the HTML version of the report to the specified directory. When neither `-noText` nor `-noHtml` is specified (the default behavior), saves both the HTML and text versions of the report to the specified directory.

Default: `summaryReport`

`-outFile` *`fileName`*

Writes the text version of the report to the specified file.

Note: You can only specify this option if you specify `-noHtml`.

Default: If you specify `-noHtml` and do not specify `-outFile`, the software writes the text version of the report to a file named `summaryReport.rpt`.

Examples

- The following command reports statics for the entire design and generates a text version of the results called `summaryReport.rpt`:

  ```
  summaryReport -noHtml -outfile summaryReport.rpt
  ```

  The following is a sample of the resulting text report:

  ```
  ============================
  General Design Information
  ============================
  Design Status: Routed
  Design Name: dtmf_chip
  # Instances: 7780
  ```

```
# Hard Macros:
    ------------------------------
    Macro Cells in Netlist
    ------------------------------
              Macro Name  Instance Count Area (um^2) Area Percentage in Core
             ram_256x16A               1      113204.162          26.353%
             rom_512x16A               1       68789.789          16.014%
                  pllclk               1       41054.915           9.557%
             ram_128x16A               1      100352.611          23.361%
# Std Cells:
    ------------------------------
    Standard Cells in Netlist
    ------------------------------
               Cell Type  Instance Count Area (um^2)
                  XOR2X4               2        113.0976
                  XOR2X2               7        232.8480
                  XOR2X1             305       8116.4160
                 XNOR2X4               8        452.3904
                 XNOR2X2               8        292.7232
                 XNOR2X1             146       3885.2352
                 TLATNX1               1         36.5904
                 SEDFFX4               8        931.3920
                 SEDFFX1             124      11136.7872
               SDFFSHQXL               1         79.8336
               SDFFSHQX4               7        745.1136
.
.
.

# Pads:
    ------------------------------
    IO Cells in Netlist
    ------------------------------
                I/O Name  Instance Count
               PCORNERDG               4
                  PDIDGZ              26
                PDO04CDG              27
                PVDD1DGZ               5
                PVSS1DGZ               5
# Net: 8641
# Special Net: 4
```

```
# IO Pins:
    ------------------------------
    Issued IO Information
    ------------------------------
    # Unplaced IO Pin 0
    # Floating IO 0
    # IO Connected to Non-IO Inst 0 53
# Pins:
    ------------------------------
    Correctness of Pin Connectivity for All Instances
    ------------------------------
    # Floating Terms 0
    # Output Term Marked Tie Hi/Lo 0
    # Output Term Shorted to PG Net 0 26897
.
.
.
```

- The following command also reports statics for the entire design, but creates both a text version and an HTML version of the results. Both versions of the report are saved in the `sumaryReport` directory. The text version is named `dtmf_chip.main.htm.ascii`.

  ```
  summaryReport -outdir summaryReport
  ```

suppressMessage

```
suppressMessage alpha_prefix numId1
    [numId2 … numIdn]
```

Suppresses error or warning messages identified by a unique ID.

Note: This command suppresses the specified message regardless of the setting specified by the `setMessageLimit` command. For example, consider the case where you have used the `setMessageLimit` command to specify that a message should be suppressed after a fixed number of occurrences. If you now run the `suppressMessage` command on this message, the message will be suppressed, irrespective of the number of times it occurs.

Parameters

alpha_prefix	Specifies the alpha prefix for the error or warning message.
numId1	Specifies the unique numerical ID of the message to suppress.
numId2 … numIdn	Specifies additional numerical IDs for the same alpha prefix.

Example

The following command suppresses the warning messages with the message ID `SI-2190`:`suppressMessage SI 2190`

This example suppresses *SOCLF-200* warning messages during design import:

```
suppressMessage SOCLF 200
```

This example suppresses *SOCLF-80 and SOCLF-200* warning messages during design import:`suppressMessage SOCLF 80 200`

Alternately, you can explicitly name each message. This example also suppresses *SOCLF-80 and SOCLF-200* warning messages during design import:

```
suppressMessage SOCLF-80 SOCLF-200
```

This example suppresses *SOCLF-80, SOCLF-200, SOCPTN-160, and SOCPTN-400* warning messages:

```
suppressMessage SOCLF 80 200 SOCPTN 160 400
```

Alternately, you can explicitly name each message. This example also suppresses SOCLF-80, SOCLF-200, SOCPTN-160, and SOCPTN-400 warning messages.

```
suppressMessage SOCLF-80 SOCLF-200 SOCPTN-160 SOCPTN-400
```

This example suppresses all messages from the *SOCLF* catalog during design import:

```
suppressMessage SOCLF
```

Multiple parameters can be used in combination on a single line. This example suppresses all *SOCLF* messages and *SOCECO-90* messages.

```
suppressMessage SOCLF SOCECO-90
```

tf_reader

```
tf_reader
     tfFileName [-layerExtMap][streamInFilename | streamOutFilename]
```

(UNIX prompt command)

Translates an Apollo Technology file into a technology file used in Encounter. This translated technology file is also required to translate (`readgds`) the GDSII data file into Cdump files. The Technology file contains the design's process and design rule technology information.

Note: To translate the Apollo design data into Encounter format, you need three files in addition to map files for cell class (`BLOCKS/IOS/STDCELLS`), cell symmetry (`X/Y/R90`), and pin use (`POWER/GROUND`). The three file are:

- The Apollo technology file
- The GDSII data file
- The stream-in or stream-out mapping file that was used to generate the GDSII data file

The template for map files can be produced by issuing the `readgds -template` command.

You can use this UNIX prompt command anytime, usually before importing the design.

Parameters

`-layerExtMap`	Generates a layer map. Dumps a LEF file, named `tfFileName.lef`, that contains LayerExtId.
`streamInFilename`	This is an Apollo mapping file that is used to load a GDS file into Apollo. This mapping file must match the GDS file that is used to load into Apollo.
`streamOutFilename`	This is an Apollo mapping file that is used to stream out a GDS file. This mapping file must match the GDS file that was streamed out of Apollo.
`tfFileName`	The Technology file name to be translated.

Example

Translates the Apollo technology file `process2k.tf` into a technology file to be used in Encounter:

```
tf_reader process2k.tf streamOut.map
```

The created files are:

- `process2k.tf.tech`

  The translated technology file for Encounter

- `process2k.tf.gdsLayerMap`

  The GDS mapping file for the `readgds` translator program

  Note: This GDS mapping file must be edited to add the correct boundary layer. This is the GDS layer number that defines the cell size.

undo

undo

Returns the design to the state it was in immediately prior to issuing the previous command. It can only be used to undo power planning, wire editing, and selected floorplanning functions.

You can use this command after you use any of the following text commands:

- `addRing`
- `addStripe`
- `alignInst`
- `connectRingPin`
- `editDelete`
- `editPowerVias`
- `flipInst`
- `redo`
- `repairStripes`
- `rotateInst`
- `shiftInst`
- `snapFPlanIO`
- `snapFPlan`
- `spaceInst`

Parameters

None

unsetMessageLimit

```
unsetMessageLimit
```

Removes the default or user-specified limit on error and warning messages. All errors and warnings are output to log.

Parameters

None

Example

The following command removes the limit on the number of error or warning messages to display:

```
unsetMessageLimit
```

unsuppressMessage

```
unsuppressMessage alpha_prefix numId1
     [numId2 … numIdn]
```

Reverses the effect of the `suppressMessage` command.

Parameters

alpha_prefix	Specifies the alpha prefix for the error or warning message.
numId1	Specifies the unique numerical ID of the message to unsuppress.
numId2 … numIdn	Specifies additional numerical IDs for the same alpha prefix.

Examples

The following command unsuppresses the warning messages with the message ID `SI-2190`:

```
unsuppressMessage SI 2190
```

This example suppresses the occurrence of the messages *SOCECO-308* and *SOCECO-310* during *optDesign -preCts*. For the rest of the flow, *SOCECO-308* is displayed but *SOCECO-310* continues to be suppressed.

```
suppressMessage SOCECO 308 310
optDesign -preCts
unsuppressMessage SOCECO-308
loadConf
```

Multiple parameters can be used in combination on a single line. This example unsuppresses all *SOCECO* messages and *SOCLF-80* messages.

```
unsuppressMessage SOCLF 80 SOCECO
```

viewLog

```
viewLog
     [-help]
     [-file logFileNmae]
```

Opens up a log file in a separate console window for viewing within the Encounter software.

The log file viewer within the Encounter software opens the current log file by default, and updates it in real time. If you specify a log file name, the software opens up the specified log file.

The log file displayed contains `[+]` and `[-]` markers that expand and collapse command information, so that you can control how much detail you want to read. Log messages are color coded for easier identification: blue messages are warnings, red messages are errors.

You can access documentation for commands in the log file that are underlined. Click on the command in the file, and the software opens an HTML window displaying the *Encounter Text Command Reference* documentation for that command.

You also can access the log file viewer from the *Tools – LogViewer* menu.

Note: You can specify `viewLog` more than once to view multiple log files in separate console windows simultaneously. However, you cannot open multiple versions of the current log file.

Parameters

`-file logFileName`	Specifies the name of the log file to open.
`-help`	Outputs a brief description that includes type and default information for each `viewLog` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man viewLog`.

Related Topics

- Outputs in the *Encounter Flat Implementation Flow Guide*

viewSnapshot

```
viewSnapshot
     [-help]
     -dir string
     [-name string]
     [-view string]
```

With this command you can open a window `SnapShot Result` for viewing saved snapshots. Snapshots are selected based on the options: `-dir` and `-name`. The window displays design views controlled by the `-view` option and also shows QoR parameters (WNS ,TNS, Placement Overlap, and congestion) saved at the time of creating the snapshot.

Parameters

`-dir` *`string`*	Specifies the directory of the saved snapshot, when opted alone it will show all snapshots saved in the directory.
`-help`	Outputs a brief description that includes type and default information for each `viewSnapshot` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man viewSnapshot`.
`-name` *`string`*	Specifies the name of snapshot to be viewed in the `SnapShot Result` window.
`-view` *`string`*	Specifies the view of the design you want to see in `SnapShot Result` window. You can see multiple views by listing them under this option. *Default:* The fplan view is shown in the `SnapShot Result` window.

Note: This command will not work in the `-nowin` mode.

Examples

- The following command will show all the snapshots present in the directory `/work/user1,` along with all QoR parameters (WNS , TNS, Placement Overlap, and Congestion). By default only the fplan view will be displayed.

  ```
  viewSnapshot -dir {/work/user1}
  ```

- The following command will show snapshots with the name `cts` along with all QoR parameters (WNS , TNS, Placement Overlap, and Congestion). By default only the fplan view will be displayed.

  ```
  viewSnapshot -dir {/work/user1} -name cts
  ```

- The following command will show snapshots with the name `cts` along with all QoR parameters (WNS , TNS, Placement Overlap, and Congestion). Only design views `cts` will be shown as opted under the -view option.

  ```
  viewSnapshot -dir {/work/user1} -name cts -view cts
  ```

- The following command will show all the snapshots present in the directory `/work/user1` along with all QoR parameters (WN , TNS, Placement Overlap, and Congestion). All the views will be shown side by side.

  ```
  viewSnapshot -dir {/work/user1} -view {fplan amoeba place}
  ```

Related Topics

For more information, see the "Tools Menu" chapter in the *Encounter Menu Reference*.

- Create Snapshot
- View Snapshot

win

```
win
```

Opens the Encounter main window. This is usually done after executing the command `encounter -init` *`tclCommandFile`*.

If you run the `win` command when the Encounter main window is already displayed, the form is refreshed and the display is fitted to the design area.

Note: If you started the Encounter software with the `encounter -nowin` command, you cannot then use the `win` command to display the Encounter main window.

Parameters

None.

writeFlowTemplate

```
writeFlowTemplate
     -help
     [-directory directory_name]
```

Copies the Encounter Foundation flows templates into the directory you specify, or to the current directory if you do not specify one with the `-directory` parameter.

Parameters

`-help`

Outputs a brief description that includes type and default information for each `writeFlowTemplate` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man writeFlowTemplate`.

`-directory directory_name`

Specifies the directory in which to save the templates.

Related Topics

- Defining Variables and Specifying Values for Timing Environment and Command Modes in the *Encounter Flat Implementation Flow Guide*
- *Encounter Foundation Flows User Guide*
- *Encounter Foundation Flows: Hierarchical Implementation Flow Guide*

zoomBox

```
zoomBox
     llx lly urx ury
```

Zooms in the viewable window to the area that encloses the specified coordinates.

Parameters

`llx`	Specifies the lower-left x coordinate for the area.
`lly`	Specifies the lower-left y coordinate for the area.
`urx`	Specifies the upper-right x coordinate for the area.
`ury`	Specifies the upper-right y coordinate for the area.

Example

The following command zooms in the viewable window to the area enclosed by the lower-left coordinates (`450 531`) and the upper-right coordinates (`550 631`)

```
zoomBox 450 531 550 631
```

zoomIn

`zoomIn`

Zooms in the viewable window.

Parameters

None

zoomOut

`zoomOut`

Zooms out from the viewable window by two times.

Parameters

None

zoomSelected

`ZoomSelected`

Zooms in the viewable window to the area that encloses the selected objects.

Parameters

None

zoomTo

`zoomTo` *`x1 y1`*

Zooms in the viewable window to the point defined by the coordinates.

Parameters

`x1 y1`	Specifies x and y coordinates of the zoom point.

Example

Zooms to point (x=100, y=200):

```
zoomTo 100 200
```

9

GUI Commands

fit

`fit`

Fits the entire design in the viewable window.

Parameters

None.

pan

```
pan x y
```

Specifies that the new center of the viewing window should shift from the current center of the viewing window by (*x*, *y*) microns.

Parameters

x y	Specifies that the design center should shift *x* microns on the x-axis and *y* microns on the y axis values from the center of the viewing window. You can specify positive or negative values for x and y. A positive value specifies shifting to the right on the x axis and shifting up on the y axis.

Example

The following example shifts the design center to the right by 200 microns and down by 300 microns from the center of the viewing window.

```
pan 200 -300
```

panCenter

```
panCenter x y
```

Pans in the viewable window to the center point defined by the specified coordinates.

Parameters

x y	Specifies the x and y coordinates of the center point.

Example

The following command pans the viewable window to the center point (x=100, y=200)

```
panCenter 100 200
```

panPage

`panPage` *`x y`*

Pans the viewable window in pages defined by the offsets. The size of each page is equal to the size of the current viewable window.

Parameters

`x y`	Specifies x and y offsets to indicate number of pages to pan. A positive value of x specifies a pan of x pages to the right. A positive number of y specifies a pan of y pages toward the top. You can specify integer values for x and y.

Examples

- The following command pans the display to the right by one page

  ```
  panPage 1 0
  ```

- The following command pans the display to the left by one page

  ```
  panPage -1 0
  ```

redraw

```
redraw
```

Refreshes the display in the viewable window. This command performs the same function as the Redraw icon (CTRL+R keyboard shortcut)

resetMultiColorsHier

```
resetMultiColorsHier
     [colorID [hInstName]]
```

Resets the color Id for all or the specified hierarchical instances.

Parameters

ColorID	Specifies the color Id to which all hierarchical instances or the specified hierarchical instance will be reset. The valid values of color Id are from 0 to 29. The color value for the color id is the same as specified in the Module Color Preferences form (accessed by clicking *Floorplan* – *Automatic* Floorplan – *Module Colors*). *Default*: If no color Id and no instance name is specified, all hierarchical instances are displayed with the color id `0`. This is the same behavior as the *Reset* button on the Module Color Preferences form.
hInstName	Specifies the name of hierarchical instance that should be displayed in the color corresponding to the specified color Id. *Default*: If no instance is specified, the color Id for all hierarchical instances is reset.

Example

- The following command resets the color Id for all hierarchical instances and their descendants to 3:

  ```
  resetMultiColorsHier 3
  ```

- The following command resets the color Id for the hierarchical instance DTMF_INST and all its descendants to the color id 3:

  ```
  resetMultiColorsHier 3 DTMF_INST
  ```

saveHInstColor

`saveHInstColor` *`fileName`*

Saves the current color setting for a hierarchical instance from the Module Color Preferences form (accessed by clicking *Floorplan – Automatic* Floorplan – *Module Colors*) to a file.

Parameters

`fileName`	Specifies the name of the file that stores 30 color settings starting with the `setLayerPreference` command. This is the same behavior as the Save button on the Module Color Preferences form.

Example

The following command saves the current color setting for a hierarchical instance from the Module Color Preferences form to a file named `hinstColorFile`:

```
saveHInstColor hinstColorFile
```

setHInstColorId

```
setHInstColorId hInstName
     ColorID
     [descendant]
```

Specifies the color ID for a hierarchical instance.

Parameters

hInstName	Name of the hierarchical instance to which this color ID applies.
ColorID	Specifies the color ID from `0` to `29`. The color value for the color id is the same as specified in the Module Color Preferences form (accessed by clicking *Floorplan – Automatic* Floorplan – *Module Colors*).
`descendant`	Specifies that the color ID should be applied to the child/ descendant instances as well. *Default*: If no ID is specified, the color id is applied only for the top-level instance.

Example

The following command applies color ID 3 to the hierarchical instance `topModule`:

```
setHInstColorId  topModule 3
```

setMultiColorsHier

```
setMultiColorsHier
     [startColorID [hInstName]]
```

Specifies the color assignment for hierarchical instances based on the color id from the Module Color Preferences form (accessed by clicking *Floorplan – Automatic* Floorplan – *Module Colors*).

Parameters

startColorID	Specifies an integer from `0` to `29` to set the starting color ID for the specified hierarchical instance. *Default*: If no ID is specified, the color ID `0` in the Module Color Preferences form is applied to the selected hierarchical instances. This is the same behavior as using the *Auto* button on the Module Color Preferences form.
hInstName	Specifies the name of the hierarchical instance whose descendants are set to the specified color ID. *Default*: if this parameter is not specified, the first hierarchical instance uses the specified color id; the second hierarchical instance uses the next color ID, and so on.

Example

The following command specifies the color Id 3 for the first hierarchical module, the color Id 4 for the second hierarchical module, and so on.

```
setMultiColorsHier 3
```

In the following command, only the descendants of `topModule` are displayed using the color ID 3.

```
setMultiColorsHier 3 topModule
```

setSelectedPtnCut

```
setSelectedPtnCut
     x1 y1 x2 y2
     PartitionName
```

Sets the area and the name for the selected partition cut area.

Parameters

x1 y1 x2 y2	Specifies the coordinates of the partition cut area.
PartitionName	Specifies the name of the partition cut area.

setSelectedPtnFeedthrough

```
setSelectedPtnFeedthrough
     x1 y1 x2 y2
     PtnFeedthroughName
     { layerID | layerIDList }
```

Sets the area, name, and layer ID for the selected partition routing feedthrough object.

Parameters

x1 y1 x2 y2	Specifies the coordinates of the routing feedthrough area.
PtnFeedthroughName	Specifies the name of the routing feedthrough.
layerID | *layerIDList*	Specifies the metal layer(s) that can route through the partition feedthrough.

Example

The following command sets the routing feedthrough object, `ptnFeethru2`, to allow routing on the M5 layer to cross over the specified area.

```
setSelectPtnFeedthrough 100.0 100.0 150.0 150.0 ptnFeedthru2 5
```

setSelectedPtnPinBlk

```
setSelectedPtnPinBlk
     x1 y1 x2 y2
     PtnPinBlkName
     { layerID | layerIDList }
```

Sets the area, name, and layer ID for the selected partition pin blockage.

Parameters

x1 y1 x2 y2	Specifies the coordinates of the partition pin blockage area.
PtnPinBlkName	Specifies the name of the partition pin blockage.
layerID|*layerIDList*	Specifies the metal layer(s) where partition pins are not to be created.

Example

The following command sets the partition pin blockage, `ptnPinBlk2`, within the specified area and on the `M4` (`Metal4`), `M5` (`Metal5`), and `M6` (`Metal6`) layers:

```
setSelectedPtnPinBlk 100.0 100.0 150.0 150.0 ptnPinBlk2 4 5 6
```

setSelectedPtnPinGuide

```
setSelectedPtnPinGuide
     x1 y1 x2 y2
     ptnPinObj
     {layerID | layerIDList}
     [cellName]
```

Sets area, name, and layer Id for the selected pin guide object.

Parameters

x1 y1 x2 y2	Specifies the coordinates of the pin guide area.
ptnPinObj	Specifies the name of the net, net group, or pin group associated with this pin guide
layerId|*layerIdList*	Specifies the metal layer(s) for the pin guide.
cellName	This parameter is required only if the object specified with the *ptnPinObj* parameter is a pin group. Specifies the name of the cell with which the pin group is associated.

setSelHInstColor

`setSelHInstColor [colorID]`

Specifies the color Id for the selected hierarchical instances.

Parameters

ColorID	Specifies the color Id to apply to all selected hierarchical instances. The color value for the color id is the same as specified in the Module Color Preferences form (accessed by clicking *Floorplan – Automatic* Floorplan – *Module Colors*) *Default*: If no Id is specified, the color Id `0` is applied to all the selected hierarchical instances.

Example

The following command applies color Id 5 to all the selected hierarchical instances:

```
setSelHInstColor 5
```

viewLast

```
viewLast
```

Displays the last view window or the previous view. This command provides the same function as the Zoom Previous icon (bindkey *w*).

windowSelect

```
windowSelect
     x1 y1 x2 y2
```

Selects objects that are enclosed by the area defined by the coordinates.

Parameters

x1 y1 x2 y2	Specifies the coordinates of the window area to enclose the objects to be selected.

Example

The following command selects objects that are totally enclosed by the area with coordinates defined as (x=10, y=200) and (x=100, y=50):

```
windowSelect 10 200 100 50
```

windowToggleSelect

```
windowToggleSelect
     x1 y1 x2 y2
```

Toggles between selecting or deselecting objects that are enclosed by the area defined by the coordinates.

Parameters

x1 y1 x2 y2	Specifies the coordinates of the window area to enclose the objects to be selected or deselected.

Example

The first command selects the objects that are totally enclosed by the area with coordinates defined as (`x1=10`, `y1=200`) and (`x2=100`, `y2=50`):

The second command deselects the objects that are totally enclosed by the same area:

```
windowToggleSelect 10 200 100 50
windowToggleSelect 10 200 100 50
```

10

Conformal Commands

checkAssembledSdcCCD

```
checkAssembledSdcCCD
     -topConstraints sdc_files
     -blockConstraints {instance1 sdc1 [instance2 sdc2] ...}
     [-clockMapFiles clock_map_files]
     [-chipNetlist netlist_files]
     [-chipConstraints sdc_files | -chipView view_name]
     [-gui]
     [-64]
     [-xl]
     [-setupOnly]
     [-outputDir directory_name]
     [-copyFiles]
     [-preVerifyInclude do_files]
```

Checks pre-assembled design top and block constraints against post-assembled design chip constraints. The Conformal Constraint Designer (CCD) checks for exceptions, clocks, unconstrained ports, and invalid SDC command syntax, and performs reference checks. You can use this command after calling the `assembleDesign` command.

To use this command, specify the path to the CCD installation before running the Encounter® software.

Note: ILMs must be in a flattened state before you run this command. To flatten ILMs, run the `flattenIlm` command.

For more information on SDC checks that CCD uses to verify SDC data, see the “SDC Rule Checks” chapter of the *Encounter Conformal Constraint Designer Reference Manual*.

Parameters

`-64`
: Specifies 64-bit CCD.

  Default: 32-bit CCD, or 64-bit if the Encounter software starts in 64-bit mode.

`-blockConstraints {instance1 sdc1 [instance2 sdc2] ...}`
: Specifies the pre-assembly block constraints. Provide one or more combinations of instance names and corresponding constraint file names.

`-chipConstraints` *`sdc_files`*

Specifies post-`assembleDesign` chip constraint files to analyze with CCD.

Note: Do not use this parameter in conjunction with `-chipView`.

Default: Existing design constraints are passed to the software. If the design has changed since it was loaded or saved, a new constraint file is written out and passed to the software.

`-chipNetlist` *`netlist_files`*

Specifies the chip netlist files.

Default: The existing design netlist is passed to the software. If the design has changed since it was loaded or saved, a new netlist is written out and passed to the software.

`-chipView` *`view_name`* Specifies the multi-mode chip-level view. If a design has multiple modes, specify the set of view constraints to pass to CCD with this parameter.

Note: Multi-mode constraints and views must already be specified.

Default: If you do not specify `-chipConstraints`, the current design SDC file is passed to the CCD software for analysis.

`-clockMapFiles` *`clock_map_files`*

Specifies the clock map files that contain hierarchically equivalent clocks.

Note: Use the clock map files previously generated when running the `deriveTimingBudget -ccd` command.

`-copyFiles` Copies all design files in the Conformal script to the Conformal run directory. Use the `-outputDir` parameter to specify the run directory.

Default: Design files are not copied to the CCD run directory.

`-gui`

Runs the CCD software in GUI mode. In this mode, the CCD software runs as a parallel job that is separate from the Encounter session—you can continue to run additional Encounter commands while CCD runs in GUI mode in parallel.

There is no exit command at the end of the CCD script when called with this parameter, so you can continue interactive debugging in the standalone Conformal GUI after completion of the CCD script.

In this mode, the Conformal log messages are not echoed to the Encounter log file. Instead, the software creates a separate Conformal log file in the CCD run directory (see the `-outputDir` parameter).

The `-gui` parameter is not available if you start the Encounter software with the `-nowin` parameter

Default: Runs the CCD software in non-GUI mode. In this mode, the CCD software exits upon completion of the CCD script and CCD log messages are echoed to the Encounter log file. In non-GUI mode, CCD is not run as a parallel job, therefore no Encounter command is executed until the CCD script has completed running.

`-outputDir` *`directory_name`*

Specifies the directory in which to generate CCD script and log files.

Default: `./checkAssembledSdcDir`.

`-setupOnly`

Generates the CCD script without starting the CCD software. Use this parameter to customize CCD run scripts.

`-topConstraints` *`sdc_files`*

Specifies the pre-assembly top level constraint files.

`-xl`

Starts the CCD software with a CCD XL license.

Default: CCD L license.

`-preVerifyInclude` *`do_files`*

Specifies a user dofile to execute as part of the dofile generated by this command. Specify this parameter to add custom settings before the CCD software enters verify mode.

Examples

The following commands check assembled SDC constraints in the CCD software against pre-assembly SDC constraints. These commands run CCD in non-GUI batch mode, and the CCD software exits on completion of the SDC checking:

```
assembleDesign \
  dtmf_chip_hier_data/signoff/dtmf_chip.enc.dat \
  results_conv_hier_data/signoff/results_conv.enc.dat \
  tdsp_core_hier_data/signoff/tdsp_core.enc.dat \
  -timingCon ${data}/${BLOCK_NAME}.sdc \
  -topFP ${WORK_DIR}/${BLOCK_NAME}_pinassign.fp
checkAssembledSdcCCD  \
  -clockMapFiles \
    PARTITION/dtmf_chip/ccd_top.clkmap \
    PARTITION/results_conv/results_conv.do \
    PARTITION/tdsp_core/tdsp_core.do \
  -topConstraints dtmf_chip_hier_data/signoff/dtmf_chip.enc.dat/dtmf_chip.pt \
  -blockConstraints  \
    DTMF_INST/RESULTS_CONV_INST \
    results_conv_hier_data/signoff/results_conv.enc.dat/results_conv.pt \
    DTMF_INST/TDSP_CORE_INST \
    tdsp_core_hier_data/signoff/tdsp_core.enc.dat/tdsp_core.pt
```

checkBudgetSdcCCD

```
checkBudgetSdcCCD
     -partitionDir partition_directory
     [-partitionNames partition_name_list]
     [-qualityChecksOnly | -hierChecksOnly]
     [-enabledSdcRules sdc_rules]
     [-hierSdcRules hier_check_sdc_rules]
     [-blocksVsChipOnly]
     [-blockConstraints {instance1 sdc1 [instance2 sdc2 ...]}
     [-clockMapFiles clk_map_files]
     [-chipNetlist netlist_files]
     [-chipConstraints sdc_files]
     [-break]
     [-gui]
     [-64]
     [-xl]
     [-setupOnly]
     [-outputDir directory_name]
     [-copyFiles]
     [-script scriptName]
     [-preVerifyInclude do_files]
```

Checks the time budget directory's top and block level constraints against their original pre-partition chip SDCs. Conformal Constraint Designer (CCD) checks for hierarchical constraint mismatches, exceptions, clocks, unconstrained ports, and invalid SDC command syntax and performs reference checks.

For more information on SDC checks that the CCD software uses to verify SDC data, see the "SDC Rule Checks" chapter of the *Encounter Conformal Constraint Designer Reference Manual*.

You can use this command after the `savePartition` command.

To use this command, specify the path to the CCD installation before running the Encounter software.

Note: ILMs must be in a flattened state before you run this command. To flatten ILMs, run the `flattenIlm` command first.

You must have previously run `deriveTimingBudget -ccd` to generate time budgets and CCD clock map files. These files are automatically detected by the `checkBudgetSdcCCD` command in the specified partition directory.

Parameters

`-64`
Runs CCD in 64-bit mode.

Default: 32-bit mode, or 64-bit if the Encounter software starts in 64-bit mode.

`-blockConstraints {`*`instance1 sdc_file1`* `[`*`instance2 sdc_file2`*`] ...}`

Specifies budgeted constraints files to use during CCD analysis. Use this parameter to override block constraints.

Provide one or more combinations of instance names and corresponding constraint file names.

Default: The software uses the constraints in the partition directory.

`-blocksVsChipOnly`
Specifies that CCD verifies only block-level partition budgeted constraints against the original pre-partition chip constraints. When you specify this parameter, CCD does not verify top-level budgeted constraints.

Note: This parameter is supported for backwards compatibility. Cadence recommends using the `-partitionNames` parameter to specify partitions to verify, instead of using this parameter.

Default: CCD verifies all budgeted constraints files (top and block constraints) against pre-partition constraints.

`-break`

Adds a “break” command after the software checks a partition constraints file. Adding breaks allows you to debug partition constraints interactively by using the SDC Rule Manager. After browsing quality checks in the SDC Rule Manager, type the following command:

```
continue
```

The CCD software then checks the next partition and breaks again, until it checks all partition constraints.

Note: This parameter is available only when you also specify the Encounter software with the `-gui` parameter. This parameter is not available if you specify the `-hierChecksOnly` parameter.

Default: CCD does not perform breaks, so interactive debug of partition constraint’s quality checks using the SDC Rule Manager is not possible. In this case, the SDC Rule Manager displays hierarchical rule check results only. However, you can examine quality checks in the individual partition reports in the following file:

`checkBudgetSdcDir rule_check.quality.`*`partitionName`*`.rpt`

`-chipConstraints` *`sdc_files`*

Specifies post-`assembleDesign` chip constraint files to analyze using CCD.

Note: This parameter cannot be used in conjunction with `-chipView`.

Default: Passes the existing design constraints to the software. If the design has changed after it was loaded or saved, a new constraint file is written out and passed to the software.

`-chipNetlist` *`netlist_files`*

Specifies chip constraints files to analyze using CCD. Use this parameter to override chip constraints.

Default: The software detects the netlist from the Encounter database.

`-clockMapFiles` *`clk_map_files`*

Specifies clock map files that contain hierarchically equivalent clocks. Use this parameter to override clock map files.

Default: The netlist is automatically detected from the Encounter database. In most cases, the clock map files are automatically detected from the Encounter database and passed to CCD without having to use this parameter.

`-copyFiles`

Copies design files in the Conformal script to the run directory. Use the `-outputDir` parameter to specify the run directory.

Default: Design files are not copied to the CCD run directory.

`-enabledSdcRules` *`sdc_rules`*

Allows specification of user-enabled or disabled rules in CCD to be used during quality rule checking of partition constraints. You can specify multiple added and deleted disabled rules.

Default: A default set of rules is defined on script initialization. The user-enabled rules are added after this default set of rules.

`-gui`

Runs the CCD software in GUI mode. This process runs as a parallel job that is separate from the Encounter session—you can continue to run additional Encounter commands while the CCD GUI mode session is running in parallel.

Because there is no exit command at the end of the CCD script when called with the `-gui` parameter, you can continue interactive debugging in the standalone Conformal GUI after completion of the CCD script.

In `-gui` mode, Conformal log messages are not echoed to the Encounter log file. The software creates a separate Conformal log file in the CCD run directory. Use the `-outputDir` parameter to specify the run directory.

Note: The `-gui` parameter is not available if you start Encounter with the `-nowin` parameter.

Default: The CCD software runs in non-GUI mode. In this mode, the CCD software exits upon completion of the CCD script and CCD log messages are echoed to the Encounter log file. In non-GUI mode, CCD does not run as a parallel job, therefore no Encounter commands are executed until the CCD script has completed.

`-hierChecksOnly` Specifies that only hierarchical rule checking is performed by the CCD software. When you use this parameter, CCD does not perform quality rule checking.

Default: CCD performs hierarchical and quality rule checking.

`-hierSdcRules` *`hier_check_sdc_rules`*

Specifies the rules to check during hierarchical rule checking.

Default: `*HIER*`

`-outputDir` *`directory_name`*

Specifies the directory in which to generate CCD script and log files.

Default: `./checkBudgetSdcDir`

`-partitionDir` *`partition_directory`*

Specifies the budget or partition directory. Time budget constraints and clock map files are automatically detected by the `checkBudgetSdcCCD` command in the specified partition directory. For example:

```
deriveTimingBudget -ccd
partition
savePartition -dir PARTITION
checkBudgetSdcCCD -partitionDir PARTITION
```

`-partitionNames` *`partition_name_list`*

Specifies the partitions whose constraints are checked against the original pre-partition chip SDCs.

Default: CCD checks all top and block level partitions under the user-specified partition directory.

`-qualityChecksOnly` Specifies that only quality rule checking for each partition is performed by the CCD software. When you use this parameter, CCD does not perform hierarchical rule checking.

Default: CCD performs hierarchical and quality rule checking.

`-script` *`scriptName`* Specifies a dofile script to pass to the CCD software for execution.

Default: A new dofile script is generated based on the `checkBudgetSdcCCD` parameters you specify and used to run the CCD software.

`-setupOnly`

Generates the CCD script but does not start the CCD software. Use this parameter to customize CCD run scripts.

Note: The name of the generated dofile script is returned by `checkBudgetSdcCCD -setupOnly` as a string.

Tip

If the CCD software runs out of memory before completing analysis, or begins to use virtual memory, it can be called standalone after exiting Encounter. By exiting Encounter, some of the system memory is freed, which can then be utilized by CCD. This procedure helps CCD avoid using swap space and might can improve run time.

Default: CCD software is launched with the generated dofile script.

`-xl`

Specifies the CCD software's XL license.

Default: CCD L license.

`-preVerifyInclude` *`do_files`*

Specifies user dofiles to execute as part of the dofile generated by this command. Use this parameter to add custom settings before the CCD software enters verify mode.

Examples

- The following commands check time budgeted generated top and block constraints under the `PARTITION` directory against their original pre-budget constraints using the CCD software. These commands run CCD in non-GUI batch mode, and the CCD software exits on completion of the SDC checking.

```
deriveTimingBudget -ccd
partition
savePartition -dir PARTITION
checkBudgetSdcCCD -partitionDir PARTITION
```

- The following example specifies that CCD should pause after checking each partition's SDCs to allow interactive debug of the block's reported quality rules with the SDC Rule Manager:

```
checkBudgetSdcCCD -partitionDir PARTITION -break
```

- The following example tells CCD to check two partitions:

```
checkBudgetSdcCCD -partitionDir PARTITION -partitionNames top block3
```

- The following command overrides constraints to be read for two partitions:

```
checkBudgetSdcCCD -partitionDir PARTITION -blockConstraints inst1 \
    inst1.constr.pt inst2 {inst2.constr.pt extra_contraints.sdc}
```

- The following command deletes the previously defined default set of rules and then sets two rule disables:

```
checkBudgetSdcCCD -partitionDir PARTITION -enabledSdcRules \
    {delete disabled rule *} {add disabled rule *CLK_DEF1 *CLK_DEF6}
```

  This command generates the following dofile:

```
    //*********************************************
    // Default enabled CCD rules
    //*********************************************
    set ccd parameter SDC_AUTO_CHECK_SEVERITY Error
    add disabled rule CCD* > /dev/null
    delete disabled rule CCD_DGN_CMB2 *DGN_CMB5 *CLK_DEF1 ..... > /dev/null
    //*********************************************
    // User enabled CCD rules
    //*********************************************
    del disabled rule *
    delete disabled rule *CLK_DEF1 *CLK_DEF6
```

- The following command specifies that only hierarchical exception rules should be checked and generates a dofile to check the following SDC rules: `*EXC_HIER*`:

```
checkBudgetSdcCCD -partitionDir PARTITION -hierSdcRules *EXC_HIER*
```

- The following example calls `checkBudgetSdcCCD` to generate a dofile without running CCD software. The dofile can then be customized and used to run CCD:

```
set dofile [checkBudgetSdcCCD -partitionDir PARTITION -setupOnly]
<Customize the dofile as required>
checkBudgetSdcCCD -script $dofile
```

checkSdcCCD

```
checkSdcCCD
     [-netlist netlist_files]
     [-constraints sdc_files | -view view_name]
     [-gui]
     [-64]
     [-xl]
     [-setupOnly]
     [-outputDir directory_name]
     [-copyFiles]
     [-preVerifyInclude do_files]
```

Checks the quality of SDC constraints for the current design using Conformal Constraint Designer (CCD). CCD checks the constraints for exceptions, clocks, unconstrained ports, SDC command syntax, and reference checks.

To use this command, specify the path to the CCD installation before running Encounter.

Note: ILMs must be in a flattened state before you run this command. To flatten ILMs, run the `flattenIlm` command.

You can use this command after loading a design with timing constraints.

Parameters

`-64`
: Runs CCD in 64-bit mode.

  Default: 32-bit CCD, or 64-bit if the Encounter software starts in 64-bit mode.

`-constraints sdc_files`
: Specifies the constraint files to pass to the software.

  Default: The existing design constraints are passed to the software. If the design has changed since it was loaded or saved, a new constraint file is written out and passed to the software.

`-copyFiles`
: Copies design files in the Conformal script to the Conformal run directory. Use the `-outputDir` parameter to specify the run directory.

  Default: Design files are not copied to the CCD run directory.

`-gui`

Runs the CCD software in GUI mode. This process runs as a parallel job that is separate from the Encounter session—you can continue to run additional Encounter commands while the CCD GUI mode session is running in parallel.

Because there is no exit command at the end of the CCD script when called with the `-gui` parameter, you can continue interactive debugging in the standalone Conformal GUI after completion of the CCD script.

In `-gui` mode, Conformal log messages are not echoed to the Encounter log file. The software creates a separate Conformal log file in the CCD run directory. Use the `-outputDir` parameter to specify the run directory.

Note: The `-gui` parameter is not available if you start Encounter with the `-nowin` parameter

Default: The CCD software runs in non-GUI mode. In this mode, the CCD software exits upon completion of the CCD script and CCD log messages are echoed to the Encounter log file. In non-GUI mode, CCD does not run as a parallel job, therefore no Encounter commands are executed until the CCD script has completed.

`-netlist` *`netlist_files`*

Specifies the netlist files.

Default: The existing design netlist is passed to the software. If the design has changed since it was loaded or saved, a new netlist is written out and passed to the software.

`-outputDir` *`directory_name`*

Specifies the directory in which to generate CCD script and log files.

Default: `./checkSdcDir`

`-setupOnly`

Generates a CCD script without starting the CCD software. Use this parameter to customize CCD run scripts.

`-view` *`view_name`*
: Specifies the multi-mode view. If a design has multiple modes, you can specify the set of view constraints to pass to Conformal with this parameter.

  Note: Multi-mode constraints and views must already be specified.

  Default: If you do not specify `-constraints`, the current design SDC file is passed to the Conformal software for analysis.

`-xl`
: Runs the CCD software with the XL license.

  Default: CCD L license.

`-preVerifyInclude` *`do_files`*
: Specifies user dofiles to execute as part of the dofile generated by this command. Use this parameter to add custom settings before the CCD software enters verify mode.

Examples

- The following command checks the quality of the constraints associated with the current design loaded in the Encounter software. This command runs CCD in non-gui batch mode, and the CCD software exits on completion of the SDC checking.

  ```
  checkSdcCCD
  ```

- The following command checks the quality of the constraints associated with the current design loaded in the Encounter software. This command runs CCD in gui mode, and the CCD software does not exit on completion of the SDC checking.

  ```
  checkSdcCCD -gui
  ```

  You can continue with interactive debugging in CCD. Control is returned immediately to the Encounter command interface after calling `checkSdcCCD -gui` as the CCD tool runs in parallel with the Encounter software.

deriveFalsePathCCD

```
deriveFalsePathCCD
     [-machineReadable]
     [-collection report_timing_collection]
     [-timingFile file_name]
     [-netlist netlist_files]
     [-constraints sdc_files | -view view_name]
     [-ccdOptions ccd_options_strings]
     [-outputFile generated_fp_file]
     [-gui]
     [-64]
     [-xl]
     [-setupOnly]
     [-outputDir directory_name]
     [-copyFiles]
     [-preVerifyInclude do_files]
     [-reportTimingOptions report_timing_options]
     [-diagnoseTruePath]
```

Performs timing report validation (TRV) by analyzing critical false paths in CCD based on Encounter-CTE timing information and constraints. Outputs a set of false paths from CCD that you can load into the Encounter software. Identifying these false paths can eliminate unnecessary netlist optimizations and improve design area and timing.

To use this command, specify the path to the CCD installation before running the Encounter software.

Note: ILMs must be flattened before you run this command. To flatten ILMs, run the `flattenIlm` command.

You can use this command after loading a design with timing constraints.

Parameters

`-64`
: Runs CCD in 64-bit mode.

  Default: 32-bit CCD, or 64-bit if the Encounter software starts in 64-bit mode.

`-ccdOptions ccd_option_strings`
: Specifies how to handle exception validation and generation.

  For a table of available strings, see CCD Option Strings on page 39.

`-collection` *`report_timing_collection`*

Specifies that a previously generated collection of timing paths is used to pass timing information to the CCD software. The timing paths contained in the collection are then passed to the CCD software in a Standard Format Timing File.

To generate the collection, use the `report_timing -collection` parameter along with your own set of reporting parameters. For example:

```
set my_collection [report_timing -collection \
  -max_points 500 -nworst 4 -max_slack 0.1]
deriveFalsePathCCD -collection $my_collection
```

Default: The `deriveFalsePathCCD` command performs timing analysis using the following command to generate a Standard Format Timing File to pass to the CCD software:

```
report_timing -max_points 5000 -nworst 4 -max_slack 0.1
```

`-constraints` *`sdc_files`*

Specifies the SDC file or files to pass to the software.

Default: The existing design constraints are passed to the software. If the design was changed since it was loaded or saved, writes out a new constraint file and passes it to the software.

`-copyFiles`

Copies all design files in the Conformal script to the Conformal run directory. Use the `-outputDir` parameter to specify the run directory.

Default: Design files are not copied to the CCD run directory.

`-diagnoseTruePath`

Instructs the software to analyze chosen timing paths and diagnose each one as true or false. Generates a report, including the justification for each diagnosis.

Default: Analyzes only the worst timing path. Use the `-reportTimingOptions` parameter to specify the timing paths to analyze.

`-gui`

Runs the CCD software in GUI mode. In GUI mode, CCD runs as a parallel job that is separate from the Encounter session—you can continue to run additional Encounter commands while the CCD GUI mode session is running in parallel.

Because there is no exit command at the end of the CCD script when called with the `-gui` parameter, you can continue interactive debugging in the standalone Conformal GUI after completion of the CCD script.

In `-gui` mode, Conformal log messages are not echoed to the Encounter log file. The software creates a separate Conformal log file in the CCD run directory. Specify the run directory with the `-outputDir` parameter.

Note: The `-gui` parameter is not available if you start Encounter with the `-nowin` parameter

Default: The CCD software runs in non-GUI mode. In this mode, CCD exits upon completion of the CCD script and CCD log messages are echoed to the Encounter log file. In non-GUI mode, CCD does not run as a parallel job, therefore no Encounter commands are executed until the CCD script has completed.

`-machineReadable`

Passes timing file information to the CCD software in machine-readable format. This means that the command uses the `report_timing -machine_readable` parameter to generate the CCD timing file. Use this parameter when a previously generated timing file was generated that is in machine readable format, for example:

```
deriveFalsePathCCD -machineReadable -timingFile \
  machine_readable_file
```

Default: The CCD standard format is used to pass timing file information to the CCD software, so the `deriveFalsePathCCD` command performs timing analysis using the following command to generate a Standard Format Timing File to pass to the CCD software:

```
report_timing -max_points 5000 -nworst 4 -max_slack 0.1
```

`-netlist` *`netlist_files`*

Specifies the netlist file or files to pass to the software.

Default: The existing design netlist is passed to the software. If the design was changed since it was loaded or saved, writes out a new netlist and passes it to the software.

`-outputDir` *`directory_name`*

Specifies the directory in which to generate the CCD script and log files.

Default: `./deriveFalsePathDir`

`-outputFile` *`generated_fp_file`*

Specifies the output false-path file. This file is generated in the directory specified by the `-outputDir` parameter.

Default: `criticalFalsePaths.sdc`.

`-reportTimingOptions` *`report_timing_options`*

Specifies the `report_timing` options this command uses when it generates the timing file information to pass to CCD software, for example:

```
deriveFalsePathCCD \
    -reportTimingOptions {-max_slack 0.0}
```

`-setupOnly` Generates the CCD run script without starting the CCD software. Use this parameter to customize CCD run scripts.

`-timingFile` *`file_name`*

Specifies a previously generated timing debug file to pass to the CCD software.

For example, to generate a machine readable timing file for maximum slack value and pass it to the CCD software use the following commands:

```
report_timing -machine_readable -max_points 300 \
  -nworst 10 -max_slack 0.20 > timing_debug.rpt
deriveFalsePathCCD-machineReadable -timingFile
  timing_debug.rpt
```

You can compress the timing report files to reduce the size of the generated timing file. For example:

```
report_timing -machine_readable -max_points 50000 \
  -nworst 4 -max_slack  0.1 > timing_debug.rpt.gz
deriveFalsePathCCD-machineReadable -timingFile
  timing_debug.rpt.gz
```

Default: The `deriveFalsePathCCD` command performs timing analysis to generate a standard format timing file to pass to the CCD software.

`-view` *`view_name`*

Specifies the multi-mode view. If a design has multiple modes, specify the set of view constraints to pass to Conformal with this parameter. This view is also used by CTE to generate the timing debug file to pass to the Conformal software.

Note: Multi-mode constraints and views must already be specified.

Default: If you do not specify the `-constraints` parameter, the current design SDC file is passed to the Conformal software for analysis.

`-xl`

Specifies the CCD software's XL license.

Default: CCD L license.

`-preVerifyInclude` *`do_files`*

Specifies user dofiles to execute as part of the dofile generated by this command. Use this parameter to add custom settings before the CCD software enters verify mode.

CCD Option Strings

The following table shows the valid strings for the `-ccdOptions` parameter.

`-STORE_FIRST_FAIL_EXC_PATH`

Specifies that the software stores the first failed exception path found during validation and stops validating the remaining paths for the exception statement. Specify this string to reduce memory usage. *This is the default*.

`-STORE_ALL_EXC_PATH`

Specifies that the software stores and validates all exception paths. Storing all exception paths requires more memory if there are many exception paths.

`-NOSDC_PATH_LIMIT [ALL | FP | MCP | TRV]`

Specifies that there is no limit for the number of paths to expand and validate for an SDC exception statement. *This is the default*.

Use `ALL` (the default) to specify that there is no limit for all exceptions or specify the limit for just false-path (`FP`), multi-cycle-path (`MCP`), or timing report validation (`TRV`) exceptions.

`-SDC_PATH_LIMIT [ALL | FP | MCP | TRV]` *`limit`*

Specifies the maximum number of paths to expand and validate for an SDC exception statement. If an exception statement reaches the maximum number, the status of the SDC validation for that statement is “path limit reached.”

Use `ALL` (the default) to specify the same limit for all exceptions, or specify the limit for just false-path (`FP`), multi-cycle-path (`MCP`), or timing report validation (`TRV`) exceptions.

`-EXPLORED_DEPTH_PATH_LIMIT_FLOW_ONLY`

Stops exception validation for the current exception statement if a path is validated with the explored depth result and moves on to the next statement. If all statements have been validated, stops validation. *This is the default*.

`-NORMAL_EXPLORED_DEPTH_PATH_LIMIT_FLOW`

Stops exception validation for the current exception statement if a path is validated with the explored depth result, and moves on to the next statement. If all statements have been validated, returns to those statements with explored depth and validates without explored depth path limit.

`-USE_DISABLED_CLOCK_OPTimization`

Specifies that the software uses disabled clock optimization for false-path validation and generation. *This is the default*.

`-NOUSE_DISABLED_CLOCK_OPTimization`

Specifies that the software does not use the disabled clock optimization for false-path validation or generation.

Note: A path whose clock is disabled is considered a false path.

`-PROVE_COMB_FP` Verifies false-path exceptions by performing combinational checks. *This is the default*.

`-PROVE_SEQ_FP` Verifies false-path exception checks sequentially.

`-PROVE_CDC_FP` Specifies that clock domain crossing paths are validated for functional FP instead of automatic pass. *This is the default*.

`-NOPROVE_CDC_FP` Specifies specifies that clock domain crossing paths are an automatic pass for FP exception check.

`-PROVE_SET_RESET_FP`

Proves functional false paths for paths to set or reset ports of flip-flops. *This is the default*.

`-NOPROVE_SET_RESET_FP`

Specifies that any path to set or reset ports of flip-flops automatically pass.

`-NOEXPAND_CDC_FP` Enables exception statement-level CDC optimization to reduce path expansion. *This is the default*.

`-EXPAND_CDC_FP` Disables exception statement-level CDC optimization by performing path expansion.

`-NOEXPAND_TIED_FP` Enables statement-level tied path optimization to reduce path expansion. *This is the default*.

`-EXPAND_TIED_FP` Disables statement-level tied path optimization by performing path expansion.

`-NOEXPAND_DISABLED_CLOCK_FP`

Enables statement-level disabled clock optimization to reduce path expansion. *This is the default*.

Note: This option is only applicable if disabled clock optimization is used.

`-EXPAND_DISABLED_CLOCK_FP`

Disables statement-level disabled clock optimization by performing path expansion.

Note: This option is only applicable if disabled clock optimization is used.

`-NOEXPAND_SET_RESET_FP`

Enables statement-level set/reset path optimization to reduce path expansion. *This is the default*.

`-EXPAND_SET_RESET_FP`

Disables statement-level set/reset path optimization by performing path expansion.

`-MCP_CHECKS` Specifies the set of MCP atomic properties (or sub checks) to validate for each MCP check. Choose any combination of the following options:

`-SRC_STB`	Specifies the source stability MCP property, where the source must be stable for at least n-1 clock cycles before the path is enabled.
`-SRC_AVL`	Specifies the source availability MCP property, where the source must be enabled on the previous n-th clock cycle before the path is enabled.
`-DEST_STB`	Specifies the destination stability MCP property, where the path must be disabled for at least n-1 clock cycles.
`-SRC_HOLD`	Specifies the source hold MCP property, where the source must hold for at least the next n-1 clock cycle.
`-DEST_HOLD`	Specifies the destination hold MCP property, where the destination must hold for at least the next n-1 clock cycle.

Or you can choose only the following with -MCP_CHECKS:

-ALL — Specifies all MCP atomic properties to validate for each MCP check.

-NOWRITE_MONITOR_LIMIT [ALL | FP | MCP | TRV]

Specifies that there is no limit in the number of simulation monitors generated for the specified exception check type.

Use ALL (the default) to specify no limit for all exceptions, or specify the limit for just false-path (FP), multi-cycle-path (MCP), or timing report validation (TRV) exceptions.

-WRITE_MONITOR_LIMIT [ALL | FP | MCP | TRV] <limit>

Specifies a limit in the number of simulation monitors for the specified exception check type.

Use ALL (the default) to specify the same limit for all exceptions, or specify the limit for just false-path (FP), multi-cycle-path (MCP), or timing report validation (TRV) exceptions.

-GENERATION — Sets the SDC generation method.

-NOTHRESHOLD — Considers all paths (no filtering).

-THRESHOLD_PERCENTage <percentage>

Instructs CCD to consider paths with logic length count greater than the <percentage> of the maximum logic length of all paths in the design. The percentage is a real number between 0 and 1.

For example, a threshold of 0.8 corresponds to considering only paths that are longer than 80 percent of the longest path.

```
set ccd option -generation -THRESHOLD_PERCENTAGE .8
```

-VALIDATE_LIMIT <limit>

Sets the limit on how much effort to consider for generation. The default limit is 100,000 candidates.

-NOVALIDATE_LIMIT

Sets no limits on the effort.

-NORENAME_SET_PIN

Specifies that when generating an asynchronous set false-path exception to the set port of the flip-flop, the pin name is /S. *This is the default.*

`-RENAME_SET_PIN <pin_name>`

Specifies the pin name for the set port of the flip-flop.

`-NORENAME_RESET_PIN`

Specifies that when generating an asynchronous reset false-path exception to the reset port of the flip-flop, the pin name is /R. *This is the default.*

`-RENAME_RESET_PIN <pin_name>`

Specifies the pin name for the reset port of the flip-flop.

`-NORENAME_DATA_PIN`

Specifies no renaming and uses the Conformal flip-flop naming convention. The pin name is /D. *This is the default.*

`-RENAME_DATA_PIN <pin_name>`

Specifies the pin name for the data pin of the RTL flip-flop primitive.

`-SET_RESET_WARN` Generates asynchronous set and reset false-paths in a more compact form. *This is the default.*

Tip

Some of the asynchronous set or reset exceptions generated with this option might have paths to the D-port of some sequential elements in the same clock domain or to primary outputs. In this case, at the end of generation, the software will issue the following warning:

```
// 1 asynchronous set/reset false path exception
extracted
// Warning: 1 asynchronous set/reset false path
exception have paths to D-port
```

In addition, the software will issue the following warning for the created SDC file:

```
# Warning: the following asynchronous set/reset FP
exception has path(s) to D-port
```

You must validate these statements.

`-NOSET_RESET_WARN` Generates asynchronous set and reset false paths, such that there are never any functional paths included in the asynchronous set and reset exceptions. This option results in safer but less compact results.

`-PARALLEL` Specifies options to be used for running parallel validation.

Note: `Ctrl-c` on the host machine will kill all the client jobs and exit.

`-LSF` Specifies running parallel validation using the Load Sharing Facility (LSF) daemon.

Choose one of the following:

`MAX_REMOTE <num>`

Specifies the maximum number of clients for running parallel jobs in LSF. The default is 8.

Note: `-LSF MAX_REMOTE 8` is the default when using `-PARALLEL` without any sub options.

`SUBMIT_COMMAND <cmd>`

Specifies the command used to submit jobs to LSF. The default is `bsub`.

`-KILL_COMMAND <cmd>`

Specifies the command used to kill jobs in LSF. The default is `bkill`.

`-SUBMIT_OPTIONS <cmd>`

Specifies any other options in the command line for LSF (for example, the queue and priorities). The default is `" "`.

`-LOG <FILE | EMAIL>`

`FILE` specifies that each client writes a log file in the current directory with name `LSF_<job_name>.log`, where `<job_name>` is assigned internally by the software.

`EMAIL` specifies that the log of each LSF job is sent by email.

`-TEST`

Checks if the environment is properly set up for LSF. The software will print a message and exit, and will not perform validation.

`-MANUAL` Specifies running shell-based parallel validation.

`-HOST <name>`

Specifies the host machine name. It is needed only for collecting the results of parallel validation either during parallel validation or after completing validation.

`-DIRECTORY` Specifies the directory to write the results and temporary files. If this option is not specified, the files are written in the current directory.

`-NOKEEP_DIR` Erases all results files at the end of the run or when interrupted. *This is the default.*

Note: If you use shell-based parallel validation without a host, the results files will not be removed.

`-KEEP_DIR` Saves the results files in case the parallel session is interrupted and you want to continue from where it stopped previously. For a fresh parallel run, remove all the results files with the following command:

```
/bin/rm -r .ccd*.
```

`-NORECOVERY_ONLY` If there are results from a previous interrupted session, parallel validation will collect the existing results and continue with parallel validation of the remaining checks. *This is the default.*

You must remove the `.ccd_scheduled*` and `.ccd_completed*` files if they exist.

`-RECOVERY_ONLY` If there are results from a previous interrupted session, parallel validation will collect the existing results and exit.

You must remove the `.ccd_scheduled*` and `.ccd_completed*` files if they exist.

Note: To use this feature with shell-based parallel validation, you must specify the host and run the parallel validation on that host.

`-CLOCK_GROUP < [-ON | -OFF] >`

Switches between using clock groups and clock domains in exception validation and SDC rule checks.

`-ON`	Uses clock groups. *This is the default.*
`-OFF`	Uses clock domains. This specifies the previous release default.

Examples

The following commands derive the current design's critical false paths using CCD and load the SDC file that is generated into the Encounter software for further analysis and optimization. The first command runs CCD in batch mode, and the CCD software exits on completion of critical false path analysis.

```
deriveFalsePathCCD -outputFile new_ccd_false_paths.sdc
loadTimingCon -incr new_ccd_false_paths.sdc
```

Related Topics

- Optimizing Timing chapter of the *Encounter User Guide*
  - Using Conformal Constraint Designer During Timing Optimization

promoteSdcCCD

```
promoteSdcCCD
     -blockConstraints instance1 sdc1 [instance2 sdc2] ...
     [-chipConstraints sdc_files | -chipView view_name]
     [-chipNetlist netlist_files]
     [-enabledSdcRules enabled_sdc_rules]
     [-outputFile generated_fp_file]
     [-gui]
     [-64]
     [-setupOnly]
     [-outputDir directory_name]
     [-copyFiles]
     [-script scriptName]
     [-preVerifyInclude user_do_files]
```

Generates top level constraints using Conformal Constraint Designer (CCD) by promoting block-level constraints to the top level and integrating them with existing chip-level constraints.

To use this command, specify the path to the CCD installation before running Encounter.

Parameters

`-64`

Starts CCD in 64-bit mode.

Default: 32-bit CCD, or 64-bit if the Encounter software starts in 64-bit mode.

`-blockConstraints {instance1 sdc1 [instance2 sdc2] ...}`

Specifies the block constraints to pass to CCD for SDC promotion. Provide one or more combinations of instance and corresponding constraint file names.

`-chipConstraints sdc_files`

Specifies the SDC chip constraint files to pass to CCD for SDC promotion.

Note: This parameter and `-chipView` cannot be used together.

Default: The existing design constraints are passed to the software. If the design has changed since it was loaded or saved, a new constraint file is written out and passed to the software.

`-chipNetlist` *`netlist_files`*

Specifies chip constraints files to analyze using CCD. Use this parameter to override chip constraints.

Default: The netlist is automatically detected from the Encounter database.

`-chipView` *`view_name`* Specifies the multi-mode chip-level view. If a design has multiple modes, you can specify the set of view constraints to pass to Conformal with this parameter.

Note: Multi-mode constraints and views must already be specified.

Default: If you do not specify `-chipConstraints`, the current design SDC file is passed to the Conformal software for analysis.

`-copyFiles` Copies all design files in the Conformal script to the Conformal run directory. Use the `-outputDir` parameter to specify the run directory.

Default: Design files are not copied to the CCD run directory.

`-enabledSdcRules` *`enabled_sdc_rules`*

Specifies user-enabled or disabled rules in CCD to be specified during CCD dofile initialization. You can define multiple added and deleted disabled rules.

Default: The following default set of rules is defined on script initialization:

```
add rule instance -def
```

The user-enabled rules are added after this default set of rules.

`-gui`

Runs the CCD software in GUI mode. In this mode, CCD runs as a parallel job that is separate from the Encounter session—you can continue to run additional Encounter commands while the CCD GUI mode session is running in parallel.

Because there is no exit command at the end of the CCD script when called with the `-gui` parameter, you can continue interactive debugging in the standalone Conformal GUI after completion of the CCD script. In this mode, Conformal log messages are not echoed to the Encounter log file; instead, the software creates a separate Conformal log file in the CCD run directory. Specify the run directory with the `-outputDir` parameter.

Note: The `-gui` parameter is not available if you start Encounter with the `-nowin` parameter.

Default: Runs the CCD software in non-GUI mode. In this mode, the CCD software exits upon completion of the CCD script and CCD log messages are echoed to the Encounter log file. In non-GUI mode, CCD is not run as a parallel job, so no Encounter commands are executed until the CCD script has completed.

`-outputFile` *`promoted_sdc_file`*

Specifies the output SDC file.

Default: `promotedChip.sdc`. This file is generated in the directory specified by the `-outputDir` parameter.

`-outputDir` *`directory_name`*

Specifies the name of the directory in which to generate CCD script and log files.

Default: `./promoteSdcDir`.

`-setupOnly`

Generates a CCD script without starting the CCD software. Use this parameter to customize CCD run scripts.

`-script` *`scriptName`*

Specifies an existing dofile script to pass to the CCD software for execution.

Default: A new dofile script is generated based on user-specified `promoteSdcCCD` parameters and used to run the CCD software.

`-preVerifyInclude` *`do_files`*

Specifies user dofiles to execute as part of the dofile generated by this command. Use this parameter to add custom settings before the CCD software enters verify mode.

Examples

- The following command generates a new set of top-level constraints based on the specified block level constraints and the current design's constraints using the CCD software. This command runs CCD in non-GUI batch mode, and the CCD software exits on completion of the SDC checking.

```
promoteSdcCCD -blockConstraints  \
  DTMF_INST/RESULTS_CONV_INST results_conv.constr.pt \
  DTMF_INST/TDSP_CORE_INST tdsp_core.constr.pt
```

- The following command generates top-level constraints using two SDC files for the `inst2` partition. This command runs CCD in GUI mode, and the CCD software does not exit on completion of the SDC checking.

```
promoteSdcCCD  -blockConstraints \
  inst1 inst1.constr.pt inst2 {inst2.constr.pt extra_contraints.sdc}
```

runCLP

```
runCLP
     [-cpf fileName]
     [-extraLib lib …]
     [-extraVlog fileName …]
     [-setupOnly]
```

Generates the following files, then runs the Conformal® Low Power Verification tool:

- Netlist (`clp_input/feclp.v`)
- CPF file (`clp_input/feclp.cpf`)
- dofile (`clp_input/feclp.tcl`)

Parameters

`-extraLib` *`lib …`*	Specifies a list of extra libraries (`-liberty`).
`-extraVlog` *`fileName …`*	Specifies a list of extra Verilog files (`-verilog`).
`-cpf` *`fileName`*	Specifies the CPF file to use as input to the verification tool. *Default:* `feclp.cpf`, generated by the Encounter software.
`-setupOnly`	Generates input files, but does not run the verification tool.

11

Hierarchical Design Commands

saveModel

```
saveModel
     [-help]
     -directory model_directory_name
     [-cts]
     [-ilm]
     [-sdf]
     [-spef]
     [-stream
          [-mapFile mapFileName]
          [-uniquifyCellName]
          [-outputMacros]
     ]
```

Saves all necessary block level design information for top level implementation. You can use this command after block implementation. The `saveModel` command saves the following data by default:

- LEF
- Timing Library
- DEF
- FE DB

Parameters

`-cts`

Generates clock tree macro models of the design. For MMMC mode, it generates a single model file *`model_root`*/model/*`design`*.ctsmodel containing data for all the views.

For non-MMMC mode, it generates models for both `setAnalysisMode -setup` and `-hold` as *`model_root`*/model/*`design`*_{setup.ctsmodel} or *`design`*_{hold.ctsmodel}

`-dir` *`model_directory_name`*

	Specifies the name of the directory where you want to write the files. You can specify an additional level of hierarchy to define the model design stage, for example:
	`results_conv_model/postRoute/...`
	`results_conv_model/signoff/...`
	Default: `MODEL`

`-help`	Outputs a brief description that includes type and default information for each `saveModel` parameter. For a detailed description of the command and all of its parameters, use the man command: `man saveModel`.
`-ilm`	Generates ILM models. It also generates hold and setup hold models as well if the design is in the post-CTS stage.
`-sdf`	Generates the delay data for each view in MMMC mode or for setup and hold modes.
`-spef`	Generates parasitic data if a design is at least routed. It invokes extraction, if not already extracted. For extraction, it honors the `default`, `detailed` or `CCE` mode user settings.
`-stream`	Generates Stream data, if the design is routed.
`-mapFile` *`mapFileName`*	Specifies the file used for layer mapping. This file is required for successful stream out, and is read by the next tool used in the design flow. For more information, see streamOut on page 194
`-uniquifyCellNames`	Creates unique cell names if there are name collisions when the software merges files. For more information, see streamOut on page 194
`-outputMacros`	Writes LEF abstract information such as LEF pin geometries and obstructions for macros. Writes one text label per port within a LEF macro pin plus one text label per shape for feedthrough pins. For more information, see streamOut on page 194

Examples

- The following command creates a directory called `cts_condor_dma` that contains the library information (`.lef` and `.lib` files) under the `library` directory, clock tree macro model information under `model` directory, delay and parasitic information, and the design data of `dma` block:

  ```
  saveModel -cts -sdf -spef -dir cts_condor_dma
  ```

12

Import and Export Commands

addCustomBox

```
addCustomBox
     layerName
     x1 y1 x2 y2
```

Adds a rectangle shape to the specified custom layer.

You can use the `addCustomBox` command at any point in the design flow.

Parameters

`layerName`	Specifies the layer for the box. You can specify any layer.
`x1 y1 x2 y2`	Specifies the coordinates of the lower-left and upper-right points of the rectangle.

Example

- The following command creates a custom box on `myLayer`:

  ```
  addCustomBox myLayer 100 100 200 200
  ```

  For information on changing the color or fill pattern of the custom box, see "Color Preferences" in "The Main Window" chapter of the *Encounter Menu Reference*.

addCustomLine

```
addCustomLine
     layerName
     x1 y1 x2 y2
```

Adds a line to the specified layer.

You can use the `addCustomLine` command at any point in the design flow.

Parameters

`layerName`	Specifies the layer. You can specify any layer.
`x1 y1 x2 y2`	Specifies the beginning and ending coordinates for the line.

addCustomText

```
addCustomText
     layerName
     "text"
     x1 y1
     height
```

Adds text to the specified custom layer.

You can use the `addCustomText` command at any point in the design flow.

Parameters

height	Specifies the height of the text.
layerName	Specifies the layer. You can specify any layer.
text	Specifies the text. Enclose the text in quotation marks.
x1 y1	Specifies the beginning coordinate of the text.

checkDesign

```
checkDesign
     { -all
     | [-io]
       [-netlist]
       [-physicalLibrary]
       [-timingLibrary]
       [-powerGround]
       [-tieHiLo]
       [-floorplan]
       [-place]
       [-danglingNet [highlight]]
     }
     [ -noText
       [-outdir directoryName]
       [-browser]
     | -noHtml
       [-outfile fileName]
     | [-outdir directoryName]
       [-browser]
     ]
```

Checks for missing or inconsistent library and design data at any stage of the design and writes the results to a text and HTML report. This command eliminates the need to perform these checks manually.

The Encounter software checks the following data:

- I/Os
- Netlist
- Physical library
- Timing library
- Power and ground pins
- Tie-high and tie-low pins
- Floorplan
- Placement

Parameters

`-all` Performs all checks.

`-browser` Opens a browser and displays the HTML version of the report.

`-danglingNet [highlight]`

Checks the design for nets that have no instance or top-level terminals, and outputs their names into the report file.

The `-danglingNet` parameter can be used only if the `-netlist` or `-all` parameter is specified.

If you specify `highlight`, the software also highlights the "dangling" nets in the main window, after the `checkDesign` command finishes.

`-floorplan` Checks the floorplan, generates error or warning messages, and reports the following information:

- Off-grid horizontal and vertical tracks
- Instances not snapped to row site
- Unplaced I/O pins
- Off Grid Power/Ground Pre-routes
- Instances not on the manufacturing grid
- Preroutes not on the manufacturing grid (error)
- Regular preroutes not on tracks

`-io` Checks I/O pads, generates warning messages, and reports the following information:

- Unplaced I/O cells
- Floating pins that belongs to I/O pads (warning)
- I/O pins connected to non-I/O instances (directly connected to core cells)
- Unplaced I/O pins
- Floating I/O pins

`-netlist`

Checks the netlist, generates error or warning messages, and reports the following information:

- Output pins tied to power/ground nets (for example, `BUFX1 U1 (.A(net1),.Y1(1'b0))`)
- Input pins floating (warning)
- Multiple driver nets (warning)

`-noHtml`

Generates only a text version of the report.

Default: Generates both an HTML and a text version of the report.

`-noText`

Generates only an HTML version of the report.

Default: Generates both an HTML and a text version of the report.

`-outDir` *`directoryName`*

When `-noText` is specified, saves the HTML version of the report to the specified directory. If neither `-noText` nor `-noHtml` is specified, saves both the HTML and text versions of the report to the specified directory.

`-outfile` *`fileName`*

Writes the text version of the report to the specified file.

Note: You can only specify this option if you specify `-noHtml`.

Default: If you specify `-noHtml` and do not specify `-outFile`, the software writes the text version of the report to a file named `summaryReport.rpt`.

`-physicalLibrary`

Checks the physical libraries and reports whether all cells have LEF views. The software generates error messages for the following conditions:

- Cells not defined in LEF
- Cells with missing dimensions
- Pins with missing direction
- Cells pins with missing geometry
- Cell dimensions are not an integer multiple of the core site dimensions

`-place`	Checks the placement, generates error or warning messages, and reports the following information: ■ Number of instances with placement violations ■ Instances placed outside of the core area ■ Instances overlapping other instances ■ Instances overlapping placement blockages
`-powerGround`	Checks power and ground connections, generates warning messages, and reports the following information: ■ Power terminals connected to ground net ■ Ground terminals connected to power net ■ Floating power and ground terminals ■ Power and ground terminals connected to non-power and ground nets.
`-tieHiLo`	Reports unconnected tie-high or tie-low terminals, and generates a warning message.
`-timingLibrary`	Checks whether the cells used in the design have been defined in the timing library. This option does not check for the presence of timing arcs. All physical cells are excluded from this check.

Command Order

Cadence recommends that you check libraries and data as follows:

- Perform I/O checking at any time. I/O problems might not impede any tool, but they might add to design problems.
- Perform netlist checking at any time after the design has been loaded.
- Perform physical library checking before floorplanning.
- Perform power/ground checking before routing and extraction, and verifying geometry and connectivity.
- Perform timing library checking before any timing-related operation (for example, timing driven placement or routing, timing optimization, clock-tree synthesis, and static timing analysis)
- Perform tie-high and tie-low checking before routing and extraction.

Examples

The following command checks the floorplan and generates a text version of the report called `checkDesign.rpt`:

```
checkDesign -floorplan -noHtml -outfile checkDesign.rpt
```

The following is a sample of the resulting text report:

```
    ============================
    Top level Floorplan Check
    ============================
    Off-Grid Horizontal Tracks: 0
    Off-Grid Vertical Tracks: 0
    Placement grid on Mfg. grid: FALSE
    User grid a multiple of Mfg. grid: FALSE
    User grid a multiple of Mfg. grid: FALSE
    Core Row grid not a multiple of Mfg. grid: 0
    Horizontal GCell Grid off Mfg. grid: 0
    Vertical GCell Grid off Mfg. grid: 0
    AreaIO rows not on core-grid: 0
        ----------------------------
        Instances not snapped to row site
        ----------------------------
        DTMF_INST/DIGIT_REG_INST/digit_out_reg_7
        BlackBoxes Off Mfg. Grid 0
        Blocks Off Mfg. Grid 0
        BlackBoxes Off placement Grid 0
        Blocks off placement Grid 0
        Instances not snapped to row site 1
    Instances not on Mfg. Grid: 0
    PrePlaced hard-macro pins not on routing grid: 0
        ----------------------------
        Unplaced Io Pins
        ----------------------------
        tdigit[7]
        tdigit[6]
        tdigit[5]
        port_pad_data_out[14]
        port_pad_data_out[13]
        port_pad_data_out[12]
        port_pad_data_out[11]
        port_pad_data_out[10]
```

```
        port_pad_data_out[9]
        port_pad_data_out[8]
        port_pad_data_out[7]
        port_pad_data_out[6]
        port_pad_data_out[5]
    .
    .
    .
    Floating/Unconnected IO Pins 0
        Unplaced Io Pins 53
    IO Pin off Mfg. grid: 0
    IO Pin outside Die area: 0
    Overlapping IO pins: 0
    IO Pin off track: 0
    Modules with off-grid Constraints: 0
    Groups with off-grid Constraints: 0
    Floating/Unconnected Ptn Pins: 0
    Partition Pin off M. Grid: 0
    Unplaced Partition Pins: 0
    Partition Pin outside Partition box: 0
    Overlapping Partition Pins: 0
    Partition Pin Off-Track: 0
    PartitionPower Domain off Grid: 0
    PreRoute not on Mfg. grid: 0
    Off Track Pre-Routes: 0
        ------------------------------
        Off Grid Power/Ground Pre-routes
        ------------------------------
        Net Rect/Polyogn/Via Off grid coordinates
        AVSS Rect (236.542500 240.355000)
        VDD Rect (424.837500 242.125000)
        Off Grid Power/Ground Pre-routes 2
```

Related Topics

- Load and Check Data in the *Encounter Flat Implementation Flow Guide*
- Load and Check Data in the *Encounter Hierarchical Implementation Flow Guide*

checkUnique

```
checkUnique [-verbose]
```

Checks the uniqueness of the netlist. Certain features, such as clock tree synthesis and IPO, require a module (a non-leaf cell) to be instantiated only once. If the netlist is unique, the result is `1`; otherwise, the result is `0`.

You can use the `checkUnique` command after importing a design.

Parameters

`-verbose`	Displays the names of modules instantiated more than once.

commitConfig

```
commitConfig
```

Applies settings specified in the current configuration file, and imports the design.

You can use the `commitConfig` command after using the `loadConfig` command. You can only use it once during a design session.

Example

- The following command loads a configuration file, `myConfig`, without committing it in the GUI or file:

  ```
  loadConfig myConfig 0
  ```

  You can then edit the configuration file as needed.

- The following command applies the settings specified in `myConfig`:

  ```
  commitConfig
  ```

defComp

```
defComp
     defFile
     [-ignoreCellPlace lef_macro_classes… [-checkCellPrefix prefix_list]]
     [-ignoreLayerFrom layer]
     [-ignoreLayerTo layer]
     [-ignoreLogicChange]
     [-placeBlockage]
     [-reportFile fileName]
     [-row]
     [-unit units_value]
     [-bump]
     [-defAsGolden]
     [-metalShapeDiff]
```

Compares a DEF file with the physical design data in the Encounter database and reports any differences.

You can use the `defComp` command after imorting a design.

Parameters

`-bump`

Compares bump placement, routing, and net assignment. The information reported includes:

- Added, moved, and deleted bumps
- Added, changed, and deleted routing
- Added, changed, and deleted net assignments

`-checkCellPrefix` *`prefix_list`*

Checks instances whose instance name begins with the specified prefix. You can use this parameter only if you also specify the `-ignoreCellPlace` parameter. If you specify more than one prefix, you must enclose the list in braces, for example, `{CORE PAD}`.
Default: All LEF `MACRO` classes specified by the `-ignoreCellPlace` parameter are ignored.

`-defAsGolden` Treats the DEF file as the golden file when comparing the DEF file to the data in the database.

By default, the software compares the DEF to the data in the database and reports the differences relative to the database. For example, `ADDINST` in the report file means the instance is not in the database, but in the DEF file.

When you specify `-defAsGolden`, the software reports the differences relative to the DEF file. Therefore, if the instance is not in the database, but is in the DEF file, `DELINST` is written in the report file.

`defFile` Specifies the name of the DEF file you want to compare with the physical design data in the database.

`-ignoreCellPlace` *`lef_macro_classes…`*

Ignores the specified cell classes, for example `{CORE PAD}`.

`-ignoreLayerFrom` *`layer`*

Ignores the specified metal or cut layer, and all metal and cut layers above. You can specify this parameter with the `-ignoreLayerTo` parameter to select a layer range. You must specify metal layers as `M1`, `M2`, `M3`, and so on, and cut layers as `V1`, `V2`, `V3`, and so on.
Default: All layers are compared.

`-ignoreLayerTo` *`layer`*

Ignores the specified metal or cut layer, and all metal and cut layers below. You can specify this parameter with the `-ignoreLayerFrom` parameter to select a layer range. You must specify metal layers as `M1`, `M2`, `M3`, and so on, and cut layers as `V1`, `V2`, `V3`, and so on.
Default: All layers are compared.

`-ignoreLogicChange`

Ignores logic changes when comparing a DEF file with the physical design data in the Encounter database.

`-metalShapeDiff` Post-processes the original report file to provide only differences on metal layers.

`-placeBlockage` Compares placement blockage and block halo equivalence.

`-reportFile` *`fileName`*

	Specifies the name of the report containing the results of the comparison. *Default:* `defPhyDiff.rpt`
`-row`	Compares row and site equivalence, name, location, and orientation.
`-unit` *`units_value`*	Uses the specified `UNITS` value (typically, the `UNITS` value in the DEF file) to scale all distance values in the report file. *Default:* Uses the current database unit (DBU) value in the database, derived from the LEF information, to scale distance values.

Example

- The following command compares data in the `myDEF` file with that in the database, and writes the results to `myResultsFile.rpt`:

  ```
  defComp myDEF -reportFile myResultsFile.rpt
  ```

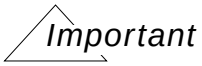
- The following command checks rows, placement blockages, and block halos in `myDEF`. This example checks all instances except class `CORE` and `PAD`; however, it checks `CORE` and `PAD` cells whose instance name prefix is `PRE`. This example also checks all wires and vias except for those on and between layers `M2` and `M6`, and writes the comparison results to `myResultsFile.def`.

  ```
  defComp -row -placeBlockage -ignoreCellPlace {CORE PAD} -checkCellPrefix PRE
  -ignoreLayerFrom M2 -ignoreLayerTo M6 -reportFile myResultsFile.def myDEF
  ```

defIn

```
defIn
     fileName
     [-allWarning]
     [-components]
     [-nets]
     [-specialnets]
     [-verilog_from_def_netlist_flow]
     [-deleteBump]
     [-deleteRDL]
     [-removeHalfWireExtensionOnPin]
```

Loads the specified DEF file.



When you load a DEF file, you must be at the same level in the hierarchy where the DEF file was saved.

The Encounter software loads the DEF file information into the database in several ways, depending on the object:

- Tracks, gcells, rows, die area

  The software deletes all of the existing objects in the database and reads in the new information.

  Note: The `defIn` command can read a rectilinear die area, but converts it to a rectangular die area with blockages at the cut off area.

- Blockages, fills, special nets

  The software compares the physical data (such as shape and layer) to the existing data in the database. If it finds the same data in the database, it ignores the data in the DEF file. If the data does not already exist, the software adds it to the database. For special nets, the software merges wire segments in the DEF file with any existing ones in the database that overlap and have the same attributes, such as layer, width, shape, and direction. The software does not delete any existing data.

- Nondefault rules, vias

  The software adds the objects to the database if they do not already exist there. If a nondefault rules definition with the same name already exists, the software ignores the one in the DEF file. If a via definition with the same name already exists, and the via is a fixed via, the software ignores the one in the DEF file. If a generated via definition with the same name already exists, the geomoetries are read, but the name might be changed. The software does not delete any existing objects of these types.

- Nets

  For each net in the DEF file, the software removes the existing regular routing (not special routing), then adds the routing from the DEF file for this net.

- Pins

  For each pin in the DEF file, the software removes the existing physical information for the pin, then adds the physical information from the DEF file for this pin. If a pin does not already exist, and it is a power or ground pin, the software adds it to the database. If the pin is not a power or ground pin, the software generates an error message.

- Groups, regions

  If a group name in the DEF file matches an existing hierarchical instance name in the database, the region's boundary constraint is attached to the hierarchical instance, overriding any existing boundary constraint. The software checks whether the group members belong to the hierarchical instance, and generates a warning message if any do not belong. If a group name does not match any hierarchical instance name, the software creates the instance group with an attached region boundary constraint, if one exists. Regions that are not used by groups in the same DEF file are ignored. The software does not remove existing groups.

  Note: You can import DEF files containing wildcards (`*`) in the `GROUPS` statement. The `defIn` command does not support multiple groups per region.

- Components

  You cannot add new logical cells. You can add new physical cells, such as:

  - Cells with `CLASS PAD`
  - Cells with `CLASS COVER` and no signal pin
  - Instance with `+ SOURCE DIST`

You can use the `defIn` command after importing a design.

You can use the `defIn -verilog_from_def_netlist_flow` command only after using the `loadDefFile` command.

Note: Do not attempt to use the `defIn -verilog_from_def_netlist_flow` command in any other Tcl scripts.

After loading the DEF netlist using the `defIn -verilog_from_def_netlist_flow` ***command, you can perform floorplanning, non-timing driven placement and routing, wire editing, and verification. You cannot use the DEF netlist flow for parasitic extraction, delay calculation, and timing-driven placement and routing effectively because the DEF names do not properly match the Verilog names used in timing constraints and timing analysis.***

Parameters

`-allWarning`	Lists all warning messages issued when you load the DEF file. *Default:* Limits the number of warning messages to the first 1,000.
`-components`	Reads the `COMPONENTS` section of the DEF file and ignores other sections, except when the `-nets` or `-specialnets` parameters are specified.
`-deleteBump`	Deletes all bumps in the database before reading the DEF file.
`-deleteRDL`	Deletes all preroutes in the database before reading the DEF file.
`fileName`	Specifies the name of the DEF file to load.
`-nets`	Reads the `NETS` section of the DEF file and ignores other sections, except when the `-components` or `-specialnets` parameters are specified.
`-removeHalfWireExtensionOnPin`	Truncates the half-width extension back to zero extension for all wire end points that overlap a pin. This is useful for DEF files created by `defOut -addHalfWireExtensionOnPin`, which forces all `NETS` section routing to use the default half-width wire extension. **Note:** Using this parameter can cause DRC violations.
`-specialnets`	Reads the `SPECIALNETS` and `VIAS` sections of the DEF file and ignores other sections, except when the `-components` or `-nets` parameters are specified
`-verilog_from_def_netlist_flow`	

Reconciles the names in the DEF and Verilog files generated by the `loadDefFile` command. This procedure is required if you want to retrieve any information that matches the original DEF file.

Refer to the procedure in "Creating a Flat Verilog Netlist from a DEF File" ***in the*** *Encounter User Guide* ***for information about the flow required for using the*** `defIn -verilog_from_def_netlist_flow` ***command. You must use the flow as documented.***

Example

- The following command loads the `TOPCHIP` DEF placement file:

```
defIn TOPCHIP.def
```

Related Topics

- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*

defOut

```
defOut
     [-floorplan]
     [-ioRow]
     [-netlist]
     [-routing]
     [-trialRoute]
     [-noSpecialNet]
     [-cutRow]
     [-noTracks]
     [-noStdCells]
     [-noCoreCells]
     [-unplaced]
     [-scanChain]
     [-selected]
     [-unit unit_per_micron]
     [-addHalfWireExtensionOnPin]
     [-verilog_from_def_netlist_flow]
     [-bumpAsPin]
     fileName
```

Writes the specified information to a DEF file. By default, the `defOut` command writes floorplan, and all placed and unplaced standard cell information to the DEF file.

The `defOut` command does not support multiple groups per region. If you run Trial Route before you run the `defOut` command, the software does not write information about `FIXED` status wires to the DEF file. The `defOut` command supports rectilinear die area.

By default, the `defOut` command does not write out any `NONDEFAULTRULE` or `VIA` defined in the LEF file. To write out a via, set the `dbgDefOutLefVias` global environment variable to `1`. To write out a nondefault rule, set the `dbgDefOutLefNDRs` global environment variable to `1`.

You can use the `defOut` command after importing a design.

Note: You can select the DEF version for your output file by setting the `dbgLefDefOutVersion` global environment variable to the version number. For example, to set the version number to 5.5, type the following command:

```
set dbgLefDefOutVersion 5.5
```

The default value is `5.7`.

Parameters

`-addHalfWireExtensionOnPin`

Typically, signal routing in the `NETS` section uses the default half-width wire extensions everywhere except for zero extensions on pins. This parameter forces the zero-extension on pins to the default half-width extension, and makes the DEF compatible for older tools that do not support DEF zero wire extensions.

Note: Using this parameter can cause DRC violations.

`-bumpAsPin` Writes bump cells as top-level pins in the `PINS` section. Use this parameter to generate a DEF file for use with QRC gate-level extraction.

Important

A DEF file generated with this parameter cannot be read back into the Encounter software.

`-cutRow` Cuts rows against placement obstructions, block halos, or power domain fence minimum gaps.

Default: Placement blockages are written to the DEF file.

`fileName` Specifies the DEF output file.

`-floorplan` Writes the floorplan data to the DEF file. This information includes rows, chip size, `CLASS PAD` instances, `CLASS BLOCK` instances, and all placed standard cells.

`-ioRow` Generates I/O rows based on I/O pad placement, and writes the I/O row information to the DEF file.

`-netlist` Writes the netlist (that is, the routing connectivity information) to the DEF file.

`-noCoreCells` Excludes core cell information from the DEF file to reduce the data size for import into APD.

`-noSpecialNet` Excludes special net information from the DEF file.

`-noStdCells` Excludes placed and unplaced standard cell information from the DEF file.

Note: If you specify this parameter with the `-floorplan` or `-unplaced` parameters, it overrides their setting.

`-noTracks`	Excludes output track and gcell statements to the DEF file.
`-routing`	Writes the routing information to the `NETS` section of the DEF file. This parameter implies the `-netlist` parameter; therefore, if you specify this parameter, you do not need to specify `-netlist`.
`-scanChain`	Writes scan chain information to the DEF file.
`-selected`	Writes only the DEF header and `COMPONENT` section for instances you have selected in the Encounter main window. If you specify the `-routing` parameter along with this parameter, the `defOut` command writes selected nets with their special routing and regular routing to the `NETS` section of the DEF file. You cannot use this parameter with any other parameters, except for the `-routing` parameter.
`-trialRoute`	Writes information about the wires generated by Trial Route to the `NETS` section of the DEF file. This option implies the `-routing` parameter; therefore, if you specify this parameter, you do not need to specify `-routing`.
`-unit` *`unit_per_micron`*	Defines the units for the DEF file.
`-unplaced`	Writes information about unplaced standard cells to the DEF file.
`-verilog_from_def_netlist_flow`	Reconciles the names in the DEF and Verilog files generated by the `loadDefFile` command. The `loadDefFile` command generates a flat Verilog file in which all of the special characters are escaped. Use this parameter to convert the escaped names back to the original names in the DEF file. For more information, see "Creating a Flat Verilog Netlist from a DEF File."

Example

- The following command writes floorplanning data, and information on placed and unplaced standard cells to `TOPCHIP_SP.def`:

  ```
  defOut -floorplan -unplaced TOPCHIP_SP.def
  ```

- The following command also writes floorplanning data, and information on placed and unplaced standard cells:

```
defOut test.def
```

- The following command writes floorplan and standard cell information to the DEF file. It does not write out unplaced standard cell information.

  ```
  defOut -floorplan
  ```

- The following command excludes all information on placed and unplaced standard cells from the DEF file:

  ```
  defOut -floorplan -noStdCells
  ```

  The resulting DEF file contains floorplan data only.

defOutBySection

```
defOutBySection
     [-compPlacement]
     [-fills]
     [-groups [-noPrintGroupRegex]]
     [-ioRow]
     [-netRouting]
     [-noComps]
     [-noNets]
     [-pins]
     [-pBlockages]
     [-rBlockages]
     [-rows]
     [-scanChains]
     [-specialNets [-noPrintWildCard]]
     [-specialNetRouting]
     [-tracks]
     [-trialRoute]
     [-unit unit_per_micron]
     [-vias]
     filename
```

Description

Writes information to specified sections in a DEF file. If you do not specify parameters for this command, it writes `COMPONENTS` and `NETS` sections.

By default, the `defOutBySection` command does not write out any via defined in the LEF file. If the `dbgDefOutLefVias` global environment variable is set to `1`, the `defOutBySection` command writes out the via defined in LEF file.

You can use the `defOutBySection` commmand after importing a design.

Parameters

`-compPlacement`	Writes component placement and orientation information to the `COMPONENTS` section of the DEF file.
`filename`	Specifies the name of the DEF output file.
`-fills`	Writes metal fill information to the `FILLS` section of the DEF file.

`-groups`	Writes group information to the `GROUPS` section of the DEF file. When you use this parameter, Encounter also writes information about regions.
`-ioRow`	Generates I/O rows based on I/O pad placement and writes the I/O row information to the DEF file.
`-netRouting`	Writes routing data to the `NETS` section of the DEF file.
`-noComps`	Excludes component information from the DEF file.
`-noNets`	Excludes net information from the DEF file. If specified, the `-netRouting` and `-trialRoute` parameters are ignored.
`-noPrintRegex`	Suppresses the use of regular expressions to represent instance names. You must also specify the `-groups` parameter.
`-noPrintWildCard`	Suppresses the use of the wildcard format to represent special net connectivity. If specified, the `-compPlacement` parameter is ignored. You must also specify the `-specialNets` parameter to use this parameter. *Default:* If you do not specify this parameter, Encounter writes connectivity information in the `SPECIALNETS` section in wildcard format if possible.
`-pins`	Writes pin information to the `PINS` section of the DEF file.
`-pBlockages`	Writes placement blockage information to the `BLOCKAGES` section of the DEF file.
`-rBlockages`	Writes routing blockage information to the `BLOCKAGES` section of the DEF file.
`-rows`	Writes row information to the `ROWS` section of the DEF file.
`-scanChains`	Writes scan chain information to the `SCANCHAINS` section of the DEF file.
`-specialNets`	Writes special net information to the `SPECIALNETS` section of the DEF file.
`-specialNetRouting`	Writes special net routing information to the `SPECIALNET` section of the DEF file to the specified file. You must also specify the `-specialNets` parameter when you use this parameter.
`-tracks`	Writes routing track information to the `TRACKS` and `GCELLGRID` statements of the DEF file.

`-trialRoute`	Writes wire information generated by the trial router to the `NETS` section of the DEF file.
`-unit` *`unit_per_micron`*	Defines the units for the DEF file.
`-vias`	Writes via information to the `VIAS` section of the DEF file.

Example

- The following command writes component placement, pin, and special net information to the `test.def` file:

  ```
  defOutBySection -compPlacement -pins -specialNets test.def
  ```

defToVerilog

```
defToVerilog
     defFile
     verilogFile
```

Converts a DEF netlist to a Verilog netlist. Encounter reads the DEF netlist, saves the netlist as Verilog, and allows you to continue your design flow without exiting the current session.

You can use the defToVerilog commnad after loading the libraries.

After loading the DEF netlist, you can perform floorplanning, non-timing driven placement and routing, wire editing, and verification. You cannot use the DEF netlist flow for parasitic extraction, delay calculation, and timing-driven placement and routing effectively because the DEF names do not properly match the Verilog names used in timing constraints and timing analysis.

Parameters

defFile	Name of the DEF file to convert.
verilogFile	Name of the Verilog file that Encounter creates.

Example

- The following example reads `myDefFile` and creates `myVerilogFile`:

  ```
  defToVerilog myDefFile myVerilogFile
  ```

deleteModule

```
deleteModule
     moduleName
     [-cell]
```

Deletes a hierarchical instance and it's module and optionally, its cell.

Parameters

`-cell`	Deletes hierarchical instance's cell.
`moduleName`	Specifies the hierarchical instance to delete.

Examples

- The following command deletes instance RESULTS_CONV_INST:
  ```
  deleteModule DTMF_INST/RESULTS_CONV_INST
  ```
- The following command deletes instance RESULTS_CONV_INST and deletes its cell:
  ```
  deleteModule DTMF_INST/RESULTS_CONV_INST -cell
  ```

disconnectDanglingPort

```
disconnectDanglingPort
     [-partitionOnly]
```

Disconnects dangling nets from ports in the whole design, from the top level to the bottom level.

You can use the `disconnectDanglingPort` command after importing a design.

Parameters

`-partitionOnly`	Disconnects dangling nets in partitions only. *Default:* Disconnects all dangling nets.

Command Order

You can use this command after you have imported the design.

Example

- The following command disconnects all dangling wires:

  ```
  disconnectDanglingPort
  ```

freeDesign

```
freeDesign
```

Removes libraries and design-specific data from the Encounter session. It can be used as a shortcut in place of exiting and re-starting Encounter.

Example

This example shows the location of `freeDesign` in a command sequence:

```
savePlace preEco.place
freeDesign
loadConf eco.conf
```

generateLef

```
generateLef
     {-lefFile lefFileName | -techFile technologyFileName -clfFile clfFileName
       | -lefFileList {technology.lef cell1.lef cell2.lef ...}}
     [-2cutVia {yes | no}]
     [-altVia {yes | no}]
     [-extraOverhangVia
       | -extraOverhangViaLengths "layer_range:overhang_value ..."]
     [-noWE]
     [-useVia {viaName | viaRuleName}]
     outputLefFileName
```

Uses input data to create an optimized LEF file with generated vias for the NanoRoute® router.

Input data can include the following:

- Original or customized LEF technology files
- Milkyway technology file
- Milkyway CLF file

Note: Include sample vias for each cut layer in the LEF file. You must also include test via rules in the input in order for the output to contain the created vias.

- Scans default vias from input data to determine the following:
  - Minimum overhang
  - Width
  - Spacing
  - End-of-line overhang
- Generates auto-vias for default, power, and other generated explicit vias.
- Retains original naming convention for LEF macro reference. If there is a conflict, utilizes original names.
- Updates technology description with all vias (for example, adds multi-cut vias automatically if a `MINIMUMCUT` statement is provided).

Use the `generateLef` command before routing with the NanoRoute router.

Parameters

`-2cutVia {yes | no}` Specifies double-cut vias.

Default: Automatically generates double-cut vias if a `MINIMUMCUT` statement is provided

`-altVia {yes | no}` Specifies alternative (hammerhead) vias.

Default: `yes`

Note: This parameter is obsolete and will be removed from the software in a future release.

`-clfFile` *`clfFileName`*

Specifies a Milkyway CLF input file.

You must also specify the `-techFile` parameter when you specify this parameter.

`-extraOverhangVia`

Generates extra-overhang vias. Use this parameter after routing, when the design is DRC-clean.

`-extraOverhangViaLengths "`*`layer_range`*`:`*`overhang_value ...`*`"`

Generate extra overhang vias on the specified layers, and of the specified lengths, only. Use this parameter after routing, when the design is DRC-clean.

Note: The specified layers are metal layers, not cut layers.

`-lefFile` *`lefFileName`*

Specifies a LEF input file.

`-lefFileList {`*`technology.lef cell1.lef cell2.lef …`*`}`

Specifies a list of LEF files to read into the software to use for generating vias with different orientations, extensions, and offsets.

Important

The file can list only one filename per line. The first file listed must be the technology LEF file.

`-noWE`

Generates a new set of vias to replace current LEF vias that do not contain built-in wire extensions.

Use this parameter to migrate to using a LEF file that does not include a `WIREEXTENSION` in the Layer (Routing) section.

Note: Define `DEFAULT` vias with adequate enclosure to meet wire extension requirements, rather than specifying the `WIREEXTENSION` in the LEF Layer (Routing) section.

When you specify the `-noWE` parameter the software does the following:

- Comments out the `WIREEXTENSION` statement in the original LEF file.
- Removes the `DEFAULT` keyword from vias that do not have enough enclosure to meet wire extension requirements.
- Generates a set of vias which have the required wire extensions in the new LEF file.

Note: This parameter is obsolete and will be removed from the software in a future release.

`outputLefFileName`

Specifies the new LEF file to output.

`-techFile` *`technologyFileName`*

Specifies a Milkyway technology file.

You must also specify the `-clfFile` parameter when you specify this parameter.

`-useVia {`*`viaName`* `|` *`viaRuleName`*`}`

Specifies a via or via rule to use as the template for creating optimized vias.

Examples

- The following command converts an old LEF file to an optimized LEF file for NanoRoute:

  ```
  generateLef -lefFile old.lef new.lef
  ```

- The following command converts an Milkyway technology file to an optimized LEF file for NanoRoute:

  ```
  generateLef -techFile mw.tf mw.lef
  ```

- The following command converts a CLF file to an optimized antenna LEF file for NanoRoute:

  ```
  generateLef -clfFile mw.clf ant.lef
  ```

generateTracks

```
generateTracks
     [-honorPitch]
     [-layerHOffset value …]
     [-layerHPitch value …]
     [-layerVOffset value …]
     [-layerVPitch value …]
     [-maxRoutingTrack value]
```

Deletes existing tracks and regenerates tracks based on the existing floorplan, signal vias, routing layer, and width and height of standard cells. Honors the `OFFSET` statement specified in the LEF Layer (Routing) section. You can specify the pitch and offset for preferred and non-preferred tracks.

The Encounter software generates tracks automatically. You do not have to run this command unless you import a DEF file using the `defIn` command and you want to overwrite the track definition in the imported file. To overwrite the track definition, run this command after running the `defIn` command.

For more information, see defIn on page 74.

Tracks for NanoRoute (whether they are generated automatically, or generated with the `generateTracks` command) are based on the following guidelines:

- The minimum pitch of the first horizontal and vertical layers matches the pitch defined in the LEF file for the layers. The pitch must be the line-to-via distance: half the wire width, plus half the via width, plus the minimum spacing on the layer.
- If the core site and height do not match with this pitch, the tool generates a warning message.
- Subsequent layers must be a 1x, 1.5x, 2x, 2.5x, 3x … multiple of the minimum pitch. For example, if the *metal3* pitch is larger than the *metal1* pitch, NanoRoute compacts the *metal3* pitch to 1x the *metal1* pitch as long as it meets the minimum pitch requirement.
- Non-preferred routing tracks of each layer are aligned with the preferred routing tracks of the next lower layer.
- Non-preferred routing tracks of the first layer are aligned with the tracks of the next upper layer.

For more information, including illustrations of line-to-via calculations, see “Modifying the Library to Achieve Optimal Pitch” in “NanoRoute Ultra Guidelines” in the *Encounter Library Development Guide*.

You can use the `generateTracks` command after importing a DEF file, and prior to running global routing.

Parameters

`-honorPitch`
: Honors the pitch specified in the LEF file. If the pitch specified in the LEF file is smaller than the minimum line-to-via pitch, adjusts the pitch to the default minimum pitch and prints a warning message in the log file. Use this parameter to spread wires to improve DFM by specifying a pitch that is larger than the minimum pitch.

`-`*`layer`*`HOffset` *`value`*
: Specifies the horizontal offset for the specified layer. You can use Encounter layer names (for example, `m1`, `m2`, `m3`) or LEF layer names.

`-`*`layer`*`HPitch` *`value`*
: Specifies the horizontal pitch for the specified layer. You can use Encounter layer names (for example, `m1`, `m2`, `m3`) or LEF layer names.

`-`*`layer`*`VOffset` *`value`*
: Specifies the vertical offset for the specified layer. You can use Encounter layer names (for example, `m1`, `m2`, `m3`) or LEF layer names.

`-`*`layer`*`VPitch` *`value`*
: Specifies the vertical pitch for the specified layer. You can use Encounter layer names (for example, `m1`, `m2`, `m3`) or LEF layer names.

`-maxRoutingTrack` *`value`*
: Specifies the highest layer to use for creating routing tracks. For example, if you specify `-maxRoutingTrack 7`, the router creates tracks on *metal1* to *metal7* only.

Examples

- The following commands generate a new routing track and run global and detailed routing:

```
generateTracks
globalDetailRoute
```

- The following command generates tracks with a horizontal offset of 0.13, a horizontal pitch of 0.26, a vertical offset of 0.14, and a vertical pitch of 0.28 on layer `m1`:

  ```
  generateTracks -m1HOffset 0.13 m1HPitch 0.26 m1VOffset 0.14 -m1VPitch 0.28
  ```

Related Topics

- Load and Check Data in the *Encounter Flat Implementation Flow Guide*
- Load and Check Data in the *Encounter Hierarchical Implementation Flow Guide*

generateVias

```
generateVias
     {-extraOverhangVia |
      -extraOverhangViaLengths "layer_range:overhang_value ..."}
```

Generates vias based on the VIARULE GENERATE statement in the LEF file. The VIARULE GENERATE statement reserves the name DEFAULT for the default routing rule for generated vias. Generated vias are stored in the Encounter database and can be shared by all applications. The generated vias are listed in the DEF VIAS section with a reference to the via rule used to derive them.

You must load a LEF file with a via rule named DEFAULT in the LEF VIARULE GENERATE section before running this command.

Parameters

`-extraOverhangVia`

Generates extra-overhang vias only. Use this parameter after routing, when the design is DRC-clean.

`-extraOverhangViaLengths "layer_range:overhang_value ..."`

Generate extra overhang vias on the specified layers, and of the specified lengths, only. Use this parameter after routing, when the design is DRC-clean.

Note: The specified layers are metal layers, not cut layers.

Example

To generate all vias, use the following command:

```
generateVias
```

To generate extra-overhang vias only, use the following command:

```
generateVias -extraOverhangVia
```

To generate extra-overhang vias on metal layers 1 through 4 with extra overhangs of 0.1 micron, and on layers 5 and 6 with extra overhangs of 0.2 microns, use the following command:

```
generateVias -extraOverhangViaLengths "1-4:0.1 5-6:0.2"
```

Related Topics

- *LEF/DEF Language Reference*
  - Via Rule Generate in the “LEF Syntax” chapter
  - Vias in the “DEF Syntax” chapter

genPinText

```
genPinText
     fileName
     {[-nets {nets…}]
     [-cells {cells…}]}
     [-layerMap fileName]
```

Generates a file containing pin text information for nets and cells. You can read this file into LVS tools such as Dracula® and Assura™ for debugging problem areas.

You can use the `genPinText` command after you have finished placing, routing, and verifying your design in Encounter.

Parameters

`-cells {`*`cells`* `…}`

Lists the hierarchical cells for with you want to generated pin text. You can use a wildcard (*) to specify all cells. You must use braces to enclose the list of cells. The *x y* values generated for the pin text are coordinates in user units local to the cell. You cannot specify this parameter with the `-nets` parameter if you want to use this file with Dracula; however, you can specify both parameters if you use Assura or a third-party LVS tool.

`fileName`

Specifies the pin text output file name. For information about the output file format see the Examples section below.

`-layerMap` *`fileName`*

Specifies the name of the file that contains the mapping between LEF layers and layers defined in the LVS tool. The layer map file has the following format:

```
LEF_LAYER_NAME VERIF_LAYER_NAME
```

If you do not specify this parameter, Encounter assumes that the LEF layer names and LVS layer names are identical.

`-nets {`*`nets`* `…}`

Lists the name of the nets for which you want to generate pin text. You cannot use a wildcard (*) to specify all nets. You must use braces to enclose the list of nets. You cannot specify this parameter with the `-cells` parameter if you want to use this file with Dracula; however, you can specify both parameters if you use Assura or a third-party LVS tool.

Examples

- The `genPinText` command creates a file containing one line of data for each cell or net your specify. The line has the following format:

  `name X=`*`x`* `Y=`*`y`* `ATTACH=`*`verif_layer_name cell_name`*

  The elements are

`name`	Specifies the net name (from the `-nets` parameter) or pin name (from the `-cell` parameter).
`X=`*`x`* `Y=`*`y`*	Specifies the x and y coordinates for the pin text.
`ATTACH=`*`verif_layer_name`*	Specifies the layer name used in the LVS tool.
`cell_name`	Specifies the cell name. The default is the top cell name.

- The following command creates a file, `pins.txt`, containing pin text information for `clk1` and `VDD`:

  ```
  getPinText -nets {clk1, VDD} pins.txt
  ```

  The output file would resemble the following:

  ```
  clk1 X=0.0 Y=10.0 ATTACH=metal1 top_cell
  VDD X=-10.0 Y=-10.0 ATTACH=metal2 top_cell
  VDD X=20.0 Y=20.0 ATTACH=metal3 top_cell
  ```

- The following command creates a file, `hedtext.txt`, for cells `block1` and `block2`:

  ```
  genPinText -cells {block1, block2} hedtext.txt
  ```

  The output file would resemble the following:

  ```
  pin1b1 X=0.0 Y=0.0 ATTACH=metal1 block1
  pin2b1 X=1.0 Y=1.0 ATTACH=metal2 block1
  ```

  and so on, until all pins for `block1` are listed. Coordinates are local to `block1`.

  ```
  pin1b2 … block2
  ```

  and so on, for all pins for `block2`.

getCmdLogFileName

`getCmdLogFileName`

Reports the name of the command log file.

You can use the `getCmdLogFileName` command at any time during an Encounter session.

getDesignMode

```
getDesignMode
     [-process]
     [-quiet]
```

Displays the following information about a specified OpenAccess mode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the OpenAccess mode parameters.

Parameter

parameter_names	Displays information for the specified parameters. See setDesignMode for descriptions of the setDesignMode parameters you can specify.
-quiet	Displays the current settings for the specified parameter in Tcl list format only. If you specify -quiet without a parameter, the software displays the current settings of all setDesignMode parameters in Tcl list format.

Examples

- The following command displays the current setting of the -process parameter:

  ```
  getDesignMode -process
  ```

  The software displays the following information:

  ```
  -process 90                # int, default=90, min=10, max=250
  ```

- The following command displays the current setting for the -process parameter in Tcl list format only:

  ```
  getDesignMode -process -quiet
  ```

The software displays the following information:

```
90
```

getExportMode

```
getExportMode
     [-fullPinout]
     [-implicitPortMapping]
     [-quiet]
```

Displays the following information about a `setExportMode` parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setExportMode` parameters.

Parameters

`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `setExportMode` for descriptions of the parameters you can specify.
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setExportMode` parameters in Tcl list format.

Examples

- The following command displays the current setting of the `-fullPinout` parameter:

  ```
  getExportMode -fullPinout
  ```

  The software displays the following information:

  ```
  -fullPinout true                # bool, default=false, user setting
  true
  ```

- The following command displays the current settings for all `setExportMode` parameters:

  ```
  getExportMode
  ```

  The software displays the following information:

  ```
  -fullPinout true                # bool, default=false, user setting
  -implicitPortMapping false      # bool, default=false
  {fullPinout true} {implicitPortMapping false}
  ```

- The following command displays the current setting of the `-fullPinout` parameter in Tcl list format only:

  ```
  getExportMode -fullPinout -quiet
  ```

  The software displays the following information:

  ```
  true
  ```

- The following command displays the current settings for all `setExportMode` parameters in Tcl list format only:

  ```
  getExportMode -quiet
  ```

  The software displays the following information:

  ```
  {fullPinout true} {implicitPortMapping false}
  ```

getImportMode

```
getImportMode
     [-bufferTieAssign]
     [-keepEmptyModule]
     [-minDbuPerMicron]
     [-quiet]
     [-syncReLativePath]
     [-timerMode]
     [-treatUndefinedCellAsBbox]
     [-useLefDef56]
     [-verticalRow]
```

Displays the following information about a specified `setImportMode` parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setImportMode` parameters.

Parameters

`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `setImportMode` for descriptions of the parameters you can specify.
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setImportMode` parameters in Tcl list format.

Examples

- The following command displays the current setting for the `-keepEmptyModule` parameter:

```
getImportMode -keepEmptyModule
```

The software displays the following information:

```
-keepEmptyModule true                # bool, default=true
true
```

- The following command displays the current settings for all `setImportMode` parameters:

  ```
  getImportMode
  ```

  The software displays the following information:

  ```
  -bufferTieAssign false               # bool, default=false
  -keepEmptyModule true                # bool, default=true
  -minDBUPerMicron 0                   # int, default=0
  -syncRelativePath false              # bool, default=false
  -timerMode false                     # bool, default=false
  -treatUndefinedCellAsBbox false      # bool, default=false
  -useLefDef56 true                    # bool, default=true
  -verticalRow false                   # bool, default=false


  {bufferTieAssign false} {keepEmptyModule true} {minDBUPerMicron 0}
  {syncRelativePath false} {timerMode false} {treatUndefinedCellAsBbox false}
  {useLefDef56 true} {verticalRow false}
  ```

- The following command displays the current setting for the `-keepEmptyModule` parameter in Tcl list format only:

  ```
  getImportMode -keepEmptyModule -quiet
  ```

  The software displays the following information:

  ```
  true
  ```

- The following command displays the current settings for all `setImportMode` parameters in Tcl list format only:

  ```
  getImportMode -quiet
  ```

  The software displays the following information:

  ```
  {bufferTieAssign false} {keepEmptyModule true} {minDBUPerMicron 0}
  {syncRelativePath false} {timerMode false} {treatUndefinedCellAsBbox false}
  {useLefDef56 true} {verticalRow false}
  ```

getLogFileName

```
getLogFileName [-fullPath]
```

Reports the name of the log file.

You can use the `getLogFileName` command at any time during an Encounter session.

Parameters

`-fullPath`	Reports the complete path to the log file.

Examples

- The following command reports the name of the log file for the current Encounter session:

  ```
  getLogFileName
  ```

  The software reports:

  ```
  encounter.log1003
  ```

- The following command reports the complete path to the log file for the current Encounter session:

  ```
  getLogFileName -fullPath
  ```

  The software reports:

  ```
  /icd/soclinx23/output/linux/07.10-e028_1/ph_46_08h34sec/encounter.log1003
  ```

getOasisOutMode

```
getOasisOutMode
     [-cblockCompression]
     [-removeNets]
     [-SEcompatible]
     [-SEvianames]
     [-snapToMGrid]
     [-specifyViaName]
     [-supportPathType4]
     [-textSize]
     [-uniquifyCellNamesPrefix]
     [-virtualConnection]
     [-quiet]
```

Displays the following information about a `setOasisOutMode` parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setOasisOutMode` parameters.

Parameters

`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `setOasisOutMode` for descriptions of the parameters you can specify.
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setOasisOutMode` parameters in Tcl list format.

Examples

- The following command displays the current setting of the `-cblockCompression` parameter:

  ```
  getOasisOutMode -cblockCompression
  ```

  The software displays the following information:

  ```
  -cblockCompression true                    # bool, default=true
  true
  ```

- The following command displays the current settings for all `setOasisOutMode` parameters:

  ```
  getOasisOutMode
  ```

  The software displays the following information:

  ```
  -cblockCompression true                    # bool, default=true
  -removeNets {}                             # string, default=""
  -SEcompatible false                        # bool, default=false
  -SEvianames false                          # bool, default=false
  -snapToMGrid false                         # bool, default=false
  -specifyViaName default                    # string, default=default
  -supportPathType4 true                     # bool, default=true
  -textSize 1                                # float, default=1
  -uniquifyCellNamesPrefix false             # bool, default=false
  -virtualConnection true                    # bool, default=true
  {cblockCompression true} {removeNets {}} {SEcompatible false} {SEvianames
  false} {snapToMGrid false} {specifyViaName default} {supportPathType4 true}
  {textSize 1} {uniquifyCellNamesPrefix false} {virtualConnection true}
  ```

- The following command displays the current setting of the `-cblockCompression` parameter in Tcl list format only:

  ```
  getOasisOutMode -cblockCompression -quiet
  ```

  The software displays the following information:

  ```
  true
  ```

- The following command displays the current settings for all `setOasisOutMode` parameters in Tcl list format only:

  ```
  getOasisOutMode -quiet
  ```

  The software displays the following information:

  ```
  {cblockCompression true} {removeNets {}} {SEcompatible false} {SEvianames
  false} {snapToMGrid false} {specifyViaName default} {supportPathType4 true}
  {textSize 1} {uniquifyCellNamesPrefix false} {virtualConnection true}
  ```

getOaxMode

```
getOaxMode
     [-cutRows]
     [-dataModel]
     [-fullLayerList]
     [-fullPath]
     [-instPlacedIfUnknown]
     [-locking]
     [-logicOnlyImport]
     [-nativeLef]
     [-quiet]
     [-saveRelativePath]
     [-updateMode]
```

Displays the following information about a specified OpenAccess mode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the OpenAccess mode parameters.

Parameter

parameter_names	Displays information for the specified parameters. You can specify one or more parameters. See setOaxMode for descriptions of the OpenAccess mode parameters you can specify.
-quiet	Displays the current settings for the specified parameters in Tcl list format only. If you specify -quiet without any parameters, the software displays the current settings of all setOaxMode parameters in Tcl list format.

Examples

- The following command displays the current setting of the -fullPath parameter:

```
getOaxMode -fullPath
```

The software displays the following information:

```
-fullPath false             # bool, default=false
false
```

- The following command displays the current settings for all `setOaxMode` parameters:

  ```
  getOaxMode
  ```

  The software displays the following information:

  ```
  -cutRows false              # bool, default=false
  -dataModel 4                # int, default=4, min=3, max=4
  -fullPath false             # bool, default=false
  -nativeLef false            # bool, default=false
  -updateMode false           # bool, default=false


  {cutRows false} {dataModel 4} {fullPath false} {nativeLef false} {updateMode
  false}
  ```

- The following command displays the current setting for the `-fullPath` parameter in Tcl list format only:

  ```
  getOaxMode -fullPath -quiet
  ```

  The software displays the following information:

  ```
  false
  ```

- The following command displays the current settings for all `setOaxMode` parameters in Tcl list format only:

  ```
  getOaxMode -quiet
  ```

  The software displays the following information:

  ```
  {cutRows false} {fullPath false} {nativeLef false} {updateMode false}
  ```

getStreamOutMode

```
getStreamOutMode
     [-removeNets]
     [-SEcompatible]
     [-SEvianames]
     [-snapToMGrid]
     [-specifyViaName]
     [-supportPathType4]
     [-textSize]
     [-uniquifyCellNamesPrefix]
     [-version]
     [-virtualConnection]
     [-quiet]
```

Displays the following information about a `setStreamOutMode` parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setStreamOutMode` parameters.

Parameters

`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `setStreamOutMode` for descriptions of the parameters you can specify.
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setStreamOutMode` parameters in Tcl list format.

Examples

- The following command displays the current setting of the `-removeNets` parameter:

```
getStreamOutMode -removeNets
```

The software displays the following information:

```
-removeNets {}                          # string, default=""
{}
```

- The following command displays the current settings for all `setStreamOutMode` parameters:

  ```
  getStreamOutMode
  ```

  The software displays the following information:

  ```
  -removeNets {}                          # string, default=""
  -SEcompatible false                     # bool, default=false
  -SEvianames false                       # bool, default=false
  -snapToMGrid false                      # bool, default=false
  -specifyViaName default                 # string, default=default
  -supportPathType4 true                  # bool, default=true
  -textSize 1                             # float, default=1
  -uniquifyCellNamesPrefix false          # bool, default=false
  -version 3                              # int, default=3, min=1, max=32767
  -virtualConnection true                 # bool, default=true

  {removeNets {}} {SEcompatible false} {SEvianames false} {snapToMGrid false} {sp
  ecifyViaName default} {supportPathType4 true} {textSize 1}
  {uniquifyCellNamesPr
  efix false} {version 3} {virtualConnection true}
  ```

- The following command displays the current setting of the `-removeNets` parameter in Tcl list format only:

  ```
  getStreamOutMode -removeNets -quiet
  ```

  The software displays the following information:

  ```
  {}
  ```

- The following command displays the current settings for all `setStreamOutMode` parameters in Tcl list format only:

  ```
  getStreamOutMode -quiet
  ```

  The software displays the following information:

  ```
  {}
  ```

lefOut

```
lefOut
     [-5.3 | -5.4 | -5.5 | -5.6 | -5.7]
     [-noCutObs]
     [-stripePin [-PGpinLayers layer_number_list]]
     [-specifyTopLayer layer_number [-extractBlockObs]]
     [-extractBlockPGPinLayers layer_number_list]
     fileName
```

Generates hierarchical design abstract (LEF) information for the current routed block-level design. By default, `lefOut` creates power and ground pins on the design, and creates cut-outs (obstructions) in the blockages for signal pins, power and ground pins, vias, and power and ground stripes. The output LEF file contains antenna information, if you run antenna verification before you run a top-down methodology using the `lefOut` command.

In a bottom-up flow (where the block was not created using the Encounter `createPartition` command), you normally create a simple LEF abstract that contains:

- one obstruction (OBS) shape per routing layer
- signal pins located along the block boundary
- power pins for the top metal layer

If the block used four routing layers out of a possible six layers, you would run the following command:

```
lefOut -noCutObs -stripePin -PGpinLayers 4 -specifyTopLayer 4
```

You can use the `lefOut` command after routing a block-level design.

Parameters

`-5.3 | -5.4 | -5.5 | -5.6 | -5.7`

Specifies the LEF version number. This value overrides any value you set with the `dbgLefDefOutVersion` variable.

Default: The software uses the current `dbgLefDefOutVersion` value. The default value for `dbgLefDefOutVersion` is `5.7`.

`-extractBlockObs`
: Extracts the obstruction (`OBS`) information from any `CLASS BLOCK` or `CLASS RING` macros with obstructions on layers that are above the specified top layer limit (`-specifyTopLayer`) and writes out the obstructions to the LEF file.

  You must specify the `-specifyTopLayer` parameter in order to use this parameter.

  Default: Only creates obstructions for layers up to `-specifyTopLayer`.

`-extractBlockPGPinLayers` *`layer_number_list`*
: Extracts the power and ground pin information from any `CLASS BLOCK` and `CLASS RING` macros with pins on layers in the specified layer list and writes out the pin information to the LEF file.

  A layer number must be an integer number that represents the metal layer number (for example, specify `4` for *metal4*). Specify a list of layer numbers by leaving a space between each number. For example, to create a list with layers *metal3*, *metal4*, and *metal5*, specify `-extBlockPGPin 3 4 5`.

  Default: Does not extract the pin information from any macros

`fileName`
: Specifies the LEF output file.

`-noCutObs`
: Creates a single obstruction shape for each routing layer.

  Default: Creates cut-outs (obstructions) in the blockages for signal pins, power and ground pins, vias, and power and ground stripes.

`-PGpinLayers` *`layer_number_list`*
: Writes out power and ground stripes on the specified layer numbers as power and ground pins.

  A layer number must be an integer number that represents the metal layer number (for example, specify `4` for *metal4*). Specify a list of layer numbers by leaving a space between each number. For example, to create a list with layers *metal3*, *metal4*, and *metal5*, specify `-PGpinLayers 3 4 5`.

  Note: You must specify the `-stripePin` parameter in order to use this parameter.

  Default: The top metal layer

`-specifyTopLayer` *`layer_number`*

Writes out obstruction information only for layers up to the specified *`layer_number`*. The layer number must be an integer number that represents the metal layer number (for example, specify `4` for *metal4*).

Default: Top routing layer number (therefore, writes out obstruction information for all layers)

`-stripePin` Writes out power and ground stripes on the top metal layer as power and ground pins. You can use this parameter in a bottom-up methodology, when you create abstracts for blocks that you will use at the next higher level of a hierarchical design.

Examples

- The following command writes a LEF file, `myLEF`, using LEF version 5.4:

  ```
  lefOut myLEF -5.4
  ```

- The following command writes power and ground stripes on layers *metal1* and *metal2* as power and ground pins in `myLEF`:

  ```
  lefOut myLef -stripePin -PGpinLayers 1 2
  ```

loadATFile

```
loadATFile {-rulefile ruleFileName | -techfile techFileName}
```

Contains the path to the technology file required by Astro. NanoRoute uses the information in the technology file to map vias and layers from LEF to a format that is compatible with Astro. It automatically generates a LEF extension file and saves it with the LEF file.

You need to run this command only once, since the mapping results are persistent.

You can use the `loadATFile` command after loading the LEF file.

Parameters

`-rulefile` *`ruleFileName`*

Specifies the path to the technology file.

`-techfile` *`techFileName`*

Specifies the technology file.

loadConfig

```
loadConfig fileName [0 | 1]
```

Loads a configuration file. If you use this command in batch mode and specify a filename, the file is loaded and the design is imported. If you specify a filename and the `0` parameter, the software loads the file, but does not import the design.

To synchronize all relative paths in the configuration file to the current working directory, you can precede the `loadConfig` command with the `setImportMode -syncRelativePath true` command. You can load this configuration file from any directory without first changing your current working directory to the previous working directory where the configuration file was saved.

You can use the `loadconfig` command to import the design when used in batch mode. You can use it only once in a design session.

For more information on the variables that can be set in the configuration file, see `Configuration File Variables`.

Parameters

`fileName`	Specifies the configuration file to load.
`[0 | 1]`	Specifies whether to apply settings in the configuration file. `1` loads the configuration file and imports the design. `0` loads the configuration file and does not import the design. *Default:* If you do not specify this parameter, `1` is selected.

Example

- The following command loads `mydesign.conf` file:

  ```
  loadConfig mydesign.conf
  ```

- The following command loads `mydesign.conf`, but does not apply the settings in the file:

  ```
  loadConfig mydesign.conf 0
  ```

  This populates the fields in the Design Import form. Click *OK* to commit settings.

Related Topics

- Load and Check Data in the *Encounter Flat Implementation Flow Guide*

- Load and Check Data in the *Encounter Hierarchical Implementation Flow Guide.*

loadDefFile

```
loadDefFile
     [-hier [-stub stubFile | -reflib {listOfRefLibs}]]
```

Reads a DEF file to build Encounter's in-memory database. You can then continue to use other Encounter commands.

After loading the DEF netlist, you can perform floorplanning, non-timing driven placement and routing, wire editing, and verification. You cannot use the DEF netlist flow for parasitic extraction, delay calculation, and timing-driven placement and routing effectively because the DEF names do not properly match the Verilog names used in timing constraints and timing analysis.

Parameters

defFile	Specifies the name of the DEF file to load.
-reflib *listOfRefLibs*	Specifies a list of OpenAccess reference libraries to load. When specified, the reference libraries are assumed to be correct by construction and verilogAnnotate is not run. This reference library is used for def2oa. Because the reference libraries are already provided, there is no need to use loadLefFile with this option.
-hier	Preserves the design hierarchy. When specified, creates a reference library by calling lef2oa. It will also create stub (stub.v) from pre-loaded LEF files. This stub will be used to run verilogAnnotate on reference libraries created by the lef2oa command. The reference library created this way is used by def2oa.
-stubs *stubfile*	Specifies the name of the stub file to load. This option can be specified only with -hier. When specified, the stub file is used in verilogAnnotate. Stub is not created internally from LEF files.

Command Order

When a list of OpenAccess reference libraries is specified, `loadLefFile` is not required before `loadDefFile`; otherwise, use this command after you load the library information.

Examples

- The following command loads the DEF file `myDef`:

  ```
  loadDefFile myDef
  ```

- The following command preserves the design hierarchy while loading the OpenAccess reference libraries and the DEF file `myDef.def`:

  ```
  loadDefFile -hier -reflib {reflib1 reflib2} mydef.def
  ```

loadLefFile

```
loadLefFile
     [-incremental]
     [-blackbox]
     [-areaIo]
     fileName
```

Imports all the technology and library data that is specified in the LEF file into the Encounter environment. In addition, this command provides the location of the LEF files the routers need to access.

Important

The `loadLefFile` command checks your LEF file to make sure that layers are defined in the correct order (as described in the *LEF/DEF Language Reference*.) For example, routing layers must be defined alternately with cut layers:

```
LAYER METAL1
TYPE ROUTING
...
END METAL1

LAYER VIA12
TYPE CUT
...
END VIA12

LAYER METAL2
TYPE ROUTING
...
END METAL2
```

If you do not define layers in the correct order (for example, if you define all routing layers before cut layers, the Encounter software displays an error message.

You can use this command after importing the design.

Parameters

`-areaIo`	Treats the pad macro cell as an areaIo cell. This option is used for versions of LEF prior to 5.5.

`-blackbox`	Treats the block macro cell as a blackbox. This option is used for versions of LEF prior to 5.5.
`fileName`	Specifies the name of LEF file.
`-incremental`	Appends the antenna data into existing layers and macro cells. This is only used for appending LEF 5.4 antenna data.

Example

Imports LEF file technology data:

```
loadLefFile sample.lef
```

loadStampModel

```
loadStampModel
     -data dataFileName
     -mod modelFileName
```

Loads stamp model files for blocks during chip assembly.

You can use the `loadStampModel` command after importing the design.

Parameters

`-data` *`dataFileName`*	Specifies the stamp model data file to load.
`-mod` *`modelFileName`*	Specifies the stamp model definition file to load.

Example

- The following command loads the `mem5.mod` stamp model definition file and the `mem5.dat` stamp model data file:

  ```
  loadStampModel -mod mem5.mod -data mem5.dat
  ```

oaCopyRestoreFiles

```
oaCopyRestoreFiles
     fromLib fromCell fromView
     toLib toCell toView
```

Copies Encounter tool-specific restore files from one OpenAccess database to another. The software copies all of the files (except `master.tag` and `layout.oa`) and renames those with the `cell` to match the `toCell` cell name. This feature supports the ECO flow, where a cellview saved with the `saveOaDesign` command and set of restore files is edited by an external tool (Virtuoso® Chip Editor), then saved to a new database to avoid overwriting the unedited version. The external tool does not copy the restore files to the new cellview.

If the originating database does not contain restore files (such as `cell.conf`, `cell.mode`, `siFix.options`, and `enc.pref.tcl`, then the software generates an error message such as the following:

```
ERROR: No restore Files found at specified source
/usr4/Tests/SOCE/OA22/PC44d_test/mydesign/top/layout
```

If the destination OpenAccess database already contains restore files, the software generates a warning message and overwrites the files.

You can use the `oaCopyRestoreFiles` command after you start a new Encounter session, and before you use the `restoreOaDesign` command.

Parameters

`fromLib fromCell fromView`

Specifies the location of the files to copy. This is the design from the previous Encounter session.

`toLib toCell toView`

Specifies where to copy the files. This refers to the new design.

Examples

- The following command copies Encounter tool-specific restore files from `myDesign/topCell/layout` to `newDesign/newTopCell/newLayout`:

  ```
  oaCopyRestoreFiles myDesign topCell layout newDesign newTopCell newLayout
  ```

- The following procedure performs ECOs with the Virtuoso Chip Editor:

  a. From an Encounter session, save the OpenAccess design.

```
saveOaDesign lib cell view
```

b. Exit the Encounter session.

c. Open the OpenAccess database in the Virtuoso Chip Editor tool and edit the design. Exit the tool.

Note: You must use either of the following: Virtuoso Chip Editor, Virtuoso Layout XL , or Virtuoso Layout GXL.

d. Start an Encounter session.

e. Copy Encounter tool-specific files from the old design to the new design.

```
oaCopyRestoreFiles fromLib fromCell fromView toLib toCell toView
```

f. Restore the OpenAccess design.

```
restoreOaDesign lib cell view
```

oaIn

```
oaIn
     lib
     cell
     view
```

Restores floorplan, placement, and routing data using the OpenAccess format. This command only reads physical information from the OpenAccess database. For information on restoring an OpenAccess database, see restoreOaDesign on page 147.

Before you use this command, you must run the `lef2oa` executable. From the `encounter` command line, type `lef2oa` without parameters, or with the `-h` parameter. You can also do this from the UNIX command line, but you must first specify the location of the `lef2oa` executable. Usage information for the `lef2oa` command is displayed.

Parameters

`cell`	Specifies the name of the top cell to import.
`lib`	Specifies the OpenAccess library name.
`view`	Specifies cell view name to import.

Example

- The following command restores data using OpenAccess:

  ```
  oaIn design amba_usb layout
  ```

oaInRC

```
oaInRC
     -oa lib cell view
     -ap analysisPointName
```

Reads SPEF-equivalent parasitic data from an OpenAccess database into memory. The OpenAccess database allows you to store different parasitic extraction results (analysis points) by name. Typical analysis point names could be `min`, `typ`, or `max`, but you can use any name. You can only read one type of parasitic data at a time, then run delay calculation or timing analysis as appropriate.

You can use the `oaInRC` command before you perform delay calculation or timing analysis.

Parameters

`-ap` *`analysisPointName`*	Selects the analysis point you want to copy.
`-oa` *`lib cell view`*	Specifies the OpenAccess database containing the parasitic data you want to copy.

Examples

- The following command reads the `min` analysis point from the `myDesign/myCell/myView` database:

  ```
  oaInRC -oa myDesign myCell myView -ap min
  ```

oaLibCreate

```
oaLibCreate
     libName
     techlibName
     [-copyTech [-libPath path]| -incrementalTech]
```

Copies or attaches a technology database to a library. If the library exists, this command does not update the library.

Note: The parameters must appear in the order specified by the syntax.

The `OaLibCreate` command is not part of the usual design flow.

Parameters

`-copyTech`	Copies the technology database specified by *techlibName* to the library specified by *libName*, and saves the library to the current working directory. *Default:* If you do not specify either `-copyTech` or `-incrementalTech`, the *techlibName* technology database is attached to the *libName* directory.
`-incrementalTech`	Creates a local technology database in the design library that can be incrementally modified. Creating a local technology database enables you to save new design-specific information to the design library technology database while still referencing a read-only library containing the base technology information. *Default:* If you do not specify either `-copyTech` or `-incrementalTech`, the *techlibName* technology database is attached to the *libName* directory.
libName	Specifies the name of a library to be created. The software creates the library and adds its name to the `lib.defs` file.
`-libPath` *path*	Specifies the name of a directory in which to save the library. The path you specify must include the directory that will become the library itself. *Default:* *./libName* (Saves the library to the current working directory)

techlibName	Specifies the name of an existing library that contains an OpenAccess technology database. If you specify `-copyTech`, you can specify only one library name. If you specify `-incrementalTech`, you can specify one library, or a Tcl list of library names.

Examples

- The following command creates the library `myLib`, and attaches the technology database in `myRefLib` to `myLib`:

  ```
  oaLibCreate myLib myRefLib
  ```

  Note: If `myRefLib` is read-only, the `saveOaDesign` command generates an error if it needs to update the technology database.

- The following command generates an error because only one reference library can be specified:

  ```
  oaLibCreate myLib {myRefLib1 myRefLib2}
  ```

- The following command creates the library `myLib`, and copies the technology database from `myRefLib` to `myLib`. If `myRefLib` is modified, the two technology databases will become out of sync. Additionally, any future write operations from the `saveOaDesign` command will modify the technology database copy in `myLib` and also cause the databases to be out of sync.

  ```
  oaLibCreate myLib myRefLib -copyTech
  ```

- The following command creates the library `myLib`, and creates a local technology database in `myLib` that references the technology databases in `myRefLib1` and `myRefLib2`. Any future write operations from the saveOaDesign command will modify the local technology database in `myLib`.

  ```
  oaLibCreate myLib {myRefLib1 myRefLib2} -incrementalTech
  ```

oaOut

```
oaOut
     lib
     cell
     view
     [-copyTechFromReflib]
     [-leafViewNames {list_of_view_names}]
     [-refLibs {list_of_libraries}]
     [-noOverwriteAbstract]
     [-noConnectivity]
     [-cutRows]
```

Saves floorplan, placement, and routing data in the OpenAccess format.

Note: The `oaOut` command does not save Trial Route information, in order to make it easier to write out a database that can be used by other routers. If you want to save Trial Route information in the OpenAccess format, use the saveOaDesign command.

Parameters

`-copyTechFromReflib`	Copies the technology file to the current design library. *Default:* If you do not specify this parameter, Encounter attaches the technology to the current design library. **Note:** This parameter applies only when you create libraries.
`-cutRows`	Specifies that rows are cut against placement obstructions and block halos, similar to the behavior of the `defOut` command. *Default:* Does not cut rows.
`-leafViewNames {`*`list_of_view names`*`}`	Specifies a list of view names to be searched for during leaf cell instance binding.

`-noConnectivity`	Writes out a design that contains instances and physical shapes. When specified, the software creates an OA module domain using the instance hierarchy, but does not attempt to create any nets, pins, or instance pins. This behavior is similar to writing a DEF file without the NETS section. *Important* Designs written out using the `-noConnectivity` parameter cannot be reloaded into the Encounter software.
`-noOverwriteAbstract`	Prevents the software from overwriting abstracts of partitions and blackboxes. *Default:* Encounter overwrites abstracts of partitions and blackboxes. Encounter never overwrites abstracts of library data.
`-refLibs {`*`list_of_libraries`*`}`	Specifies the names of the external libraries that will contain the leaf cell masters, LEF abstract, and technology file. This parameter only accepts library names, not paths. The mapping between the external library name and its path is defined in the `lib.defs` file. Encounter references technology information only from the first library in the list of libraries.
`cell`	Specifies the target cell name.
`lib`	Specifies the target OpenAccess library name.
`view`	Specifies the target view name.

Example

- The following command saves data to OpenAccess format:

```
oaOut design amba_usb layout -refLibs {stdLib blockLib ioLib}
  -copyTechFromReflib
  -leafViewNames {abstract}
```

oasisOut

```
oasisOut
     oasisFile
     [-attachInstanceName attributeNumber]
     [-attachNetName attributeNumber]
     [-dieAreaAsBoundary]
     [-libName libraryName]
     [-mapFile mapFile]
     [-merge {listOfExternalOASISFiles}]
     [-uniquifyCellNames]
     [-mode {ALL | FILLONLY | NOFILL | NOINSTANCES}]
     [-offset x y]
     [-outputMacros]
     [-stripes numberOfStripes]
     [-structureName {structureName | -noStructureName}]
     [-units {100 | 200 | 1000 | 2000 | 10000 | 20000}]
```

Creates an OASIS file of the current database. Writes a summary of errors and warnings to the Encounter log file.

- If you do not specify a map file, the Encounter software creates a generic map file, in OASIS format. The generic file shows the mapping that was used, if none was specified. You must customize the file to make it appropriate for your design.
- You must specify all layers to include in the file. Layers that are not specified in the map file are not included in the file created by this command.
- If you do not specify `-outputMacros` , `LEFPIN` and `LEFOBS` do not apply. If you specify `-outputMacros`, and `LEFPIN` and `LEFOBS` are not specified in the map file for the layers in the LEF macros, the OASIS structures for those macros will be empty.

Parameters

`-attachInstanceName` *`attributeNumber`*

Specifies the instance name value added to the attribute on SREF of the specified attribute number.

`-attachNetName` *`attributeNumber`*

Specifies that the net name will be added to the specified attribute number on the wiring shapes.

`oasisFile`

Specifies the OASIS output file. Include the `.gz` extension to enable file compression (for example, `oasis_file.oasis.gz`).

`-dieAreaAsBoundary`

Forces the `oasisOut` command to write the die area as a polygon (that is, a boundary). Otherwise, the command writes the die area as a rectangle (that is, a box).

`-libName` *`libraryName`*

Specifies the library to convert to OASIS format.
Default: `DesignLib`

`-mapFile` *`mapFile`*

Specifies the file used for layer mapping. This file is required for successful generation of an OASIS file, and is read by the next tool used in the design flow.

Important

If you do not specify a file with this parameter, the Encounter software will read an `oasisOut.map` file if one exists. If an `oasisOut.map` file does not exist, the software creates a generic map file that you must customize for your design.

For information on the map file format, see About the GDSII Stream or OASIS Map File in the *Encounter User Guide*.

`-merge {`*`listOfExternalOASISFiles`*`}`

Specifies a single file or list of files to merge. Checks all merged files for name collisions, and generates a warning if any names are changed. Uses the following conventions for new names:

- Replaces dot separators (`.`) with underscores (_). For example, `test.test_OASIS.OASIS` is renamed `test_test_OASIS_OASIS`.
- Adds unique identifiers. For example, if more than one cell is named `test.test.OASIS.OASIS`, they are renamed `test_test_OASIS_OASIS_1`, `test_test_OASIS_OASIS_2`, and so on.

All merge files must be OASIS files. Compressed files are acceptable (include the `.gz` extension to enable file compression). The Encounter software automatically creates blackboxes when merging and writing macro files. It ignores any cells in the merge files that are not used in the design.

If you specify more than one file, separate the file names with spaces and enclose the list of file names in braces ({}) or double quotation marks (")

You can use a wildcard to read all OASIS files in a specified directory. The usage is as follows:

```
streamOut -merge OASISdir/*.oasis
```

where *`OASISdir`* is the name of the directory that contains the `.oasis` files.

Use caution when using wildcards, as they might affect the order in which the software reads merge files.

For more information and detailed examples, see Merging GDSII Stream or OASIS Files in the "Importing and Exporting Designs" chapter of the *Encounter User Guide*.

`-mode {ALL | FILLONLY | NOFILL | NOINSTANCES}`

Identifies the layers to write.

Specify one of the following:

- `ALL`—Writes all layer information specified in `mapFile` as follows:
  - All instances
  - All via instances
  - All generated via cells
- `FILLONLY`—Writes only fill layers specified in `mapFile` as follows:
  - No instances
  - Only fill via instances
  - Only generated fill via cells
- `NOFILL`—Writes only non-fill layers specified in `mapFile`.
- `NOINSTANCES`—Writes wiring (including vias) only; does not write out `COMPONENT` instances.

`-noStructureName`

Enables you merge to files without loading an Encounter database when the `-merge` parameter is also specified.

`-offset` *`x y`*

Specifies, in microns, the location of the design in the GDSII output file. When the GDSII output file is saved, the design shifts to the specified location.

`-outputMacros`

Writes LEF abstract information such as LEF pin geometries and obstructions for macros. Writes one text label per port within a LEF macro pin plus one text label per shape for feedthrough pins.

Specify this parameter to create GDSII output that contains the LEF macro structures as well as the design data. You can also use this parameter to debug LEF macro definitions against verification DRC rules. The output contains macro pins and obstructions.

Note: `LEFPIN` and `LEFOBS` apply only when you specify this parameter. If you specify this parameter and `LEFPIN` and `LEFOBS` are not specified in the map file for the layers in the LEF macros the OASIS structures for those macros will be empty.

`-stripes` *`numberOfStripes`*

Specifies the number of stripes that divide the design.
Default: `1`

For example, setting the `-stripes` parameter to 2 divides the design into two equal windows, so the Encounter software creates three files. Setting the `-stripes` parameter to 3 divides the design into three equal windows, so the Encounter software creates four files. All polygons and shapes whose lower-left corner is in the current window are written with that stripe.

If the `-stripes` parameter value is greater than `1`, the naming conventions are as follows:

- `OASIS_file.oasis` (contains the top structure, structure reference to top structures of other stripes, and via definitions)
- `GDS_file.oasis.1` (contains stripe number 1, `structureName.1`)
- `GDS_file.oasis.2` (contains stripe number 2, `structureName.2`)

`-structureName` *`structureName`*

Specifies the superstructure that consists of all geometries that the map file writes out.
Default: Current design's top cell name

`-uniquifyCellNames`

Creates unique cell names if there are name collisions when the Encounter software merges files.

Important

You must specify the `-merge` parameter if you specify this parameter. When you list the merge files, list the top-level file first, as the software uses the first name in the search path when renaming cells.

When a cell name is repeated, the software renames the cell *`cellName_filename`*, where *`filename`* is the name of the merge file in which the cell name appears.

The Encounter software changes the name of references to that cell in the merge file. The software changes the reference name if the referenced cell is defined in the same file, regardless of whether the cell is defined before or after the reference. If a reference in a merge file refers to a cell definition in a different merge file, then the reference is not changed in the OASIS output file.
Default: If you do not specify this parameter, the Encounter software ignores duplicate cell names.

For more information and detailed examples, see Merging GDSII Stream or OASIS Files in the "Importing and Exporting Designs" chapter of the *Encounter User Guide*.

`-units {100 | 200 | 1000 | 2000 | 10000 | 20000}`

Specifies the resolution for values in the OASIS file. Choose `100`, `200`, `1000`, `2000`, `10000`, or `20000`.
Default: Units specified in LEF file

Related Topics

- Importing and Exporting Designs chapter in the *Encounter User Guide*
  - "Converting an Encounter Database to GDSII Stream or OASIS Format"
  - "GDSII Stream or OASIS Map File Format"

pdefIn

`pdefIn` *`fileName`*

Loads the Physical Design Exchange Format (PDEF) placement file. Encounter automatically reads global routing information from the PDEF file.

You can use the `pdefIn` command after importing the design.

When you load a PDEF file, you must be at the same level in the hierarchy where the PDEF file was saved.

Parameters

`fileName`	Specifies the PDEF placement file.

Example

- The following command loads the PDEF placement file `TOPCHIP_SP.pdef`:

```
pdefIn TOPCHIP_SP.pdef
```

replaceLefMacro

```
replaceLefMacro
     [-macros {listOfMacros}]
     lefFile
```

Replaces the current MACRO information in the design with the MACRO information in the specified LEF file. You can use the `replaceLefMacro` command to change the geometry information of a macro, such as boundary size, pin location, pin layer, and pin size. You cannot use the command to change the number or name of pins for a macro.

The `replaceLefMacro` command only reads and replaces the MACRO information from the LEF file; it does not use any other information in the file.

Parameters

`lefFile`
Specifies the LEF file to read.

`-macros {`*`listOfMacros`*`}`
Replaces the information for the specified macros only.

Default: Replaces the information for all macros in the LEF file.

pdefOut

```
pdefOut
     {-version 2 [-floorplan] [-noOrient] fileName}
     | {[-version 3] [-coreSite coreSiteName] [-floorplan] fileName}
     [[-routing | -grouting] fileName]
```

Writes the floorplan and placement information to a PDEF file.

Note: The `-routing` and `-grouting` parameters are mutually exclusive. If you specify both parameters, Encounter uses the second parameter you specify.

You can use the `pdefOut` command after creating a floorplan, or running placement.

Parameters

`-coreSite` *`coreSiteName`*
: Writes out the specified core site name.

`fileName`
: Specifies the name of the PDEF file.

`-grouting`
: Includes global routing information when you write a PDEF file. The `-grouting` and `-routing` parameters are mutually exclusive. If you include both in error, Encounter accepts the last one you specify.

`-floorplan`
: Specifies that only floorplan information be written into the output PDEF file. If you do not specify this option, both floorplanning and placement information appear in the output PDEF file.

`-noOrient`
: Excludes object orientation information from the output.

`-routing`
: Includes regular routing information when you write a PDEF file. The `-grouting` and `-routing` parameters are mutually exclusive. If you include both in error, Encounter accepts the last one you specify.

`-version 2`
: Outputs the PDEF file in PDEF version 2.
Default: Version 3.

`-version 3`
: Outputs the PDEF file in PDEF version 3.

Examples

- The following command writes the floorplan and placement information to the PDEF file `TOPCHIP_SP.pdef:`

  ```
  pdefOut TOPCHIP_SP.pdef
  ```

- The following command writes the floorplan information to the PDEF file `FLOORPLAN.pdef`:

  ```
  pdefOut -floorplan FLOORPLAN.pdef
  ```

- The following command writes global routing information:

  ```
  pdefOut -grouting groute.pdef
  ```

- The following command writes regular routing information:

  ```
  pdefOut -routing route.pdef
  ```

restoreDesign

```
restoreDesign
     sessionName.dat
     topCell
```

Loads the saved design files of a previous design session from the specified `.dat` directory. The files included depend on the work completed in the design session when it was saved. Files in the directory can include configuration files for design import, floorplan files, special route files, placement files, trial route files, and power switch state files.

Note: You must rebuild the timing graph after you restore the design. Run extraction, then timing analysis.

You must specify all pin types in the LEF file. Without these definitions, the Encounter software attempts to determine the pin type from the pin name when restoring a design, and does not always do this correctly. If this is the case, Encounter generates the following error message:

```
**ERROR: Current net list does not match wires file.
```

Caution

Modifying the contents of the `sessionName.dat` ***directory created by*** `saveDesign` ***can cause major unexpected results. Never modify the*** `design.dat` ***directory contents.***

You can use the `restoreDesign` command after starting the Encounter software.

Parameters

`sessionName.dat`	Specifies the `.dat` directory from which to load the files. If the session name of the saved design includes the extension `.enc`, make sure that you include this extension in the *`sessionName`*, for example, `session1.enc.dat`.
`topCell`	Specifies the top cell of the design saved in the *`sessionName`*`.dat` directory.

Examples

- The following command loads the configuration files from the `attempt_3.enc.dat` directory and imports the design:

  ```
  restoreDesign attempt_3.enc.dat TOP
  ```

The top cell in the `attempt_3.enc.dat` directory is `TOP`.

restoreOaDesign

```
restoreOaDesign
     lib
     cell
     view
```

Restores all design information from the previous design session, using the OpenAccess database format.

In order to restore a design in OpenAccess format, you must have saved the design using the saveOaDesign command in the previous session.

> **Note:** To ensure that parameterized cells (PCells) and other custom objects are restored correctly in the Encounter database, specify the following command before loading the design:
>
> ```
> setOaxMode -updateMode true
> ```
>
> For more information on this command see "setOaxMode" on page 184

Parameters

cell	Specifies the name of the top cell to restore.
lib	Specifies the name of the OpenAccess library where the design has been saved. When you save a design in OpenAccess format, the library name you specify (along with the path to the library) is saved in a file called `libs.def`. When you restore a design that was saved in OpenAccess format, the software checks the `libs.def` file for the library, to find the library's location.
view	Specifies the name of the cell view to restore.

Examples

- The following steps restore view FP in cell TOP in library myDesign from Session 1, in Session 2:

  Session 1

  1. Import the design
  2. After floorplanning, save the design in view FP.

     ```
     saveOaDesign myDesign TOP FP
     ```

3. After placement, save the design in view PL.

   ```
   saveOaDesign myDesign TOP PL
   ```

4. After routing, save the design.

   ```
   saveOaDesign myDesign TOP RT
   ```

5. Exit the design.

   ```
   exit
   ```

   Session 2

➤ Restore the FP floorplan view from cell TOP in library myDesign.

```
restoreOaDesign myDesign TOP FP
```

saveConfig

`saveConfig` *`fileName`*

Saves the design settings from the Design Import form to a configuration file. If you want to restore these design settings in another session, you can use the `restoreDesign` command to load the saved file.

You can use the `saveConfig` command after using the `commitConfig` command, or after loading a design using the Design Import form.

Parameters

`fileName`	Specifies the name of the configuration file in which to save the design import information.

Example

- The following command saves the design import information to a configuration file named `MY.conf`:

```
saveConfig MY.conf
```

saveDesign

```
saveDesign
     sessionName
     [-compress]
     [-def]
     [-netlist]
     [-tcon]
     [-rc]
     [-excludeUnusedInst]
     [-relativePath]
     [-timingGraph]
```

Saves the files for design import, floorplan, placement, routing, and power switch state information in the specified directory. You can save a design to the same location from which you restored the design. The new data, including the netlist and constraints, overwrite the previous data.

Modifying the contents of the `sessionName.dat` ***directory created by*** `saveDesign` ***can cause major unexpected results. Never modify the*** `design.dat` ***directory contents.***

You can use the `saveDesign` command at any time after importing a design.

Note: This command will issue an error message if you try to save a mulitple-suppy voltage (MSV) design in native Encounter format. Instead, create a Commmon Power Format (CPF) file containing CPF-based MSV commands and load the file, or use the following commands in this specific order to save your native Encounter MSV design in a CPF database:

- `saveCPF`
- `loadCPF`
- `commitCPF`
- `saveDesign`

For backward compatibility, if you must save the MSV design in a native Encounter database, set the following variable before saving the design:

```
setVar dbgSaveOldMsvDBOnly
```

Cadence does not recommend that you do this: use the `saveCPF`, `commitCPF`, `loadCPF`, `saveDesign` flow instead.

Parameters

`-compress`

Compresses output files when you save a design. For large designs compressing Verilog, DEF, and floorplan files saves disk space and improves performance. Only files with .v, .fp, .fp.spr, .fp.pg, and .def extension are compressed.

`-def` — Saves a DEF file in addition to saving place and route files when you save the design.

`-excludeUnusedInst` — Saves a virtual partition database.

`-netlist` — Creates a netlist file even if the netlist from the current Encounter session contains no changes.

`-rc` — Creates RC extraction data, including data for incremental extraction. You must specify the `-rc` parameter to save RC extraction data. The Encounter software does not automatically save RC extraction data, even if the design was previously extracted.

Note the following limitations before using this option:

- The `-rc` parameter does not save RC extraction data if the design is not extracted or if you extract the data using the default extraction engine.
- The previously saved RC data can be restored on systems that share the same platform and architecture only. For example, if you saved the RC data on the Linux platform you cannot restore it on the IBM AIX platform. Similarly, RC data saved on a 64-bit system cannot be restored on a 32-bit system.

`-relativePath` — Defines the current working directory ($cwd) in the configuration file as "`.`".

`sessionName` — Represents the user-defined name of the current Encounter session. The recommended file extension is `.enc`. The associated data directory is *`sessionName`*`.enc.dat`.

`-tcon` — Creates a timing constraint file even if the timing constraints file from the current Encounter session contains no changes.

`-timingGraph`	Saves the timing graph information. When specified, the software saves the timing graph information to a file that it gzips, and saves the resulting `.tgz` file in the `saveDir` directory. This compression occurs automatically to reduce the amount of disk space consumed by `saveDesign`. When you load a design with the `restoreDesign` command, the software automatically loads this timing graph information, if the file exists in the `saveDir` directory. Once the design is restored, the software immediately can respond to timing queries.

Example

Copies the current design import configuration, floorplan, placement, and route files into directory `attempt_3.enc`:

```
saveDesign attempt_3.enc
```

Related Topics

- Load and Check Data in the *Encounter Flat Implementation Flow Guide*
- Load and Check Data in the *Encounter Hierarchical Implementation Flow Guide*
- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Partitions in the *Encounter Hierarchical Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Top-Level in the *Encounter Hierarchical Implementation Flow Guide*
- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- SI Closure in the *Encounter Flat Implementation Flow Guide*
- Partition SI Closure in the *Encounter Hierarchical Implementation Flow Guide*
- Top-Level SI Closure in the *Encounter Hierarchical Implementation Flow Guide*

saveDesignForPrevRelease

`saveDesignForPrevRelease` *`directoryName`*

Saves a design so that it can be loaded into a previous version of the Encounter software.

Important

Certain files (such as the mode file) do not have version control and might cause warning or error messages, or fail to load during restoreDesign. In such cases, manually modify the files.

You can use the `saveDesignForPrevRelease` command at any time after importing the design.

Parameters

`directoryName`	Specifies the directory in which to save the design.

saveNetlist

```
saveNetlist
     fileName
     [-excludeLeafCell
      | -onlyLeafCell
      | -onlyStdCell
      | -onlyMacro
      | -module moduleName]
     [-flat]
     [-phys]
     [-includePowerGround]
     [-includePhysicalInst]
     [-excludeTopCellPGPort portName]
     [-includeBumpCell]
     [-includePhysicalCell cellName]
     [-excludeLogicalCell cellName | -excludeCellInst cellName]
     [-lineLength characters_per_line]
```

Writes a netlist file of the design. The netlist is in hierarchical Verilog format only if the original netlist loaded was hierarchical. The Encounter software saves the netlist from the top level of the hierarchy to the bottom level of the hierarchy if the original netlist is specified from top to bottom. If you run this command after using `loadDefFile`, it bundles bus pins to create busses in Verilog. Saving the netlist is done after running placement with timing driven option, clock tree synthesis, scan group re-order, or IPO.

You can use this command after importing the design.

Parameters

`-excludeCellInst` *`cellName`*

Excludes the instances of the specified cells from the netlist. You can exclude logical or physical instances.

The *`cellName`* can be a single cell name or a list of cell names separated by a space and surrounded by braces (`{}`) or double quotation marks (`""`).

For example: `-excludeCellInst {cell1 cell2}` or `"cell1 cell2"`.

`-excludeLeafCell` Writes I/O instances, macro or block instances, and standard cell instances to the netlist, but does not include leaf cell definitions in the netlist.

`-excludeLogicalCell` *`cellName`*

Note: Use the `-excludeCellInst` parameter instead because it more accurately describes the function being performed. The `-excludeLogicalCell` parameter name remains for backward compatibility with existing scripts only. If you specify `-excludeLogicalCell`, the `saveNetlist` command internally calls `-excludeCellInst`.

`-excludeTopCellPGPort` *`portName`*

Excludes specified power and ground port(s) from the top cell's module port list.

The *`portName`* can be a single port name or a list of port names separated by a space and surrounded by braces (`{}`) or double quotation marks (`""`).

For example: `-excludeTopCellPGPort {VSS VDD}` or `"VSS VDD"`.

`fileName` Specifies the name of the file where the netlist is written.

`-flat` Writes a flattened Verilog netlist.

Default: If you omit this parameter, and if the original netlist loaded was hierarchical, the netlist is written in hierarchical Verilog format.

`-includeBumpCell` Writes bump cell information to the netlist.

Note: This parameter is effective only when used with either the `-phys` or `-flat` parameters.

`-includePhysicalCell` *`cellName`*

Creates a netlist file with the specified physical cells. Instances of the specified cell(s) are included, not the module definitions. The *`cellName`* can be a single cell name or a list of cell names separated by a space and surrounded by braces (`{}`) or double quotes (`""`), for example, `-includePhysicalCell {cell1 cell2}` or `"cell1 cell2"`.

Default: If you omit this parameter, no physical cell module definitions will be in the netlist file.

`-includePowerGround` Includes power and ground connections in the netlist file.

`-includePhysicalInst`

Includes physical instances, such as fillers.

`-lineLength` *`characters_per_line`*

Specifies the maximum number of characters per line for module port lists and instance port connections. When a line length exceeds the specified value, the software wraps the line at the next opportunity. You must specify a non-negative integer for the value.

`Default:` `0` (The software wraps lines at the earliest opportunity.)

`-module` *`moduleName`*

Saves the specified module of the netlist.

`-onlyLeafCell` Writes only leaf cell definitions in the netlist.

Default: If you do not specify this parameter, the software writes leaf cell definitions, standard cell instances, and macro/block instances in the netlist file.

`-onlyMacro` Writes only macro or block cell definitions to the netlist.

Default: If you do not specify this parameter, the software writes leaf cell definitions, standard cell instances, and macro/block instances in the netlist file.

`-onlyStdCell` Writes only standard cell instances in the netlist.

Default: If you do not specify this parameter, the software writes leaf cell definitions, standard cell instances, and macro/block instances in the netlist file.

`-phys` Writes out physical cell instances, and inserts power and ground nets in the netlist. This is used for LVS and for designs with multiple supply voltages.

Example

Writes out the netlist in Verilog format the current state of the design in Encounter to a file `mydesign_cts.v`:

```
saveNetlist mydesign_cts.v
```

saveOaBlackboxes

```
saveOaBlackboxes
     -lib libName
     -cell list_of_cells
     [-view viewName]
```

Saves the specified blackbox abstracts, using the OpenAccess database format.

Parameters

`-cell` *`list_of_cells`*

Specifies one or more blackboxes to save. If you specify more than one blackbox, you must enclose them in either double quotation marks or curly braces, and separate them with an empty space.

`-lib` *`libName`*

Specifies the OpenAccess library to use to save the blackbox abstracts. If an OpenAccess library with the same name already exists, the software creates a cell view in that library.

If the library does not exist, the software creates the library in the working directory, with the library name and the path name being the same. If a new library is created, the technology file is copied from the technology library to the new library.

`-view` *`viewName`*

Specifies the cell view to save.

Default: Uses the view name `abstract`

Examples

- The following command saves the blackbox `myBlackbox` in view `abstract`:

  ```
  saveOaBlackboxes -lib myLib -cell myBlackbox -view abstract
  ```

- The following command saves the blackboxes `myBlackbox1` and `myBlackbox2` in view `abstract`:

  ```
  saveOaBlackboxes -lib myLib -cell “myBlackbox1 myBlackbox2” -view abstract
  ```

- The following command also saves the blackboxes `myBlackbox1` and `myBlackbox2` in view `abstract`:

  ```
  saveOaBlackboxes -lib myLib -cell {myBlackbox1 myBlackbox2} -view abstract
  ```

saveOaDesign

```
saveOaDesign
     lib
     cell
     [view]
     [-timingGraph]
```

Saves all design information in the current design session, using the OpenAccess database format.

> **Note:** To ensure that the Encounter-modified portion of an OpenAccess database is saved without making changes to parameterized cells (PCells) or other custom objects, specify the following command before loading the design:
>
> ```
> setOaxMode -updateMode true
> ```
>
> For more information on this command, see "setOaxMode" on page 184.

To save a design in OpenAccess format, you must have done one of the following:

- Restored the design using the `restoreOaDesign` command.

or

- Imported the design using the following steps:

  a. Select *Design Import* from the Design menu.

  b. On the Design Import – Advanced – OpenAccess page, specify the *Reference Libraries* and *Abstract View Names*.

Parameters

cell	Specifies the name of the top cell to save. *Default:* The current cell name.

lib	Specifies the OpenAccess library used to save the design. If an OpenAccess library with the same name already exists, the software creates a cell view in that library. If the library does not exist, the software creates the library in the working directory, with the library name and the path name being the same. If a new library is created, the technology file is copied from the technology library to the new library. If the library you specify does not exist, but a directory exists with the same name, the software cannot create the library. When you save a design in OpenAccess format, the library name you specify (along with the path to the library) is saved in a file called `libs.def`. When you restore a design that was saved in OpenAccess format, the software checks the `libs.def` file for the library, to find the library's location.
`-timingGraph`	Saves the timing graph information to a file called *`cell`*`.tg`, in the *`lib/cell/view`* directory. When you load a design with the `restoreOaDesign` command, the software automatically loads this timing graph information, if the file exists in the directory. Once the design is restored, the software immediately can respond to timing queries.
view	Specifies the cell view to save. *Default:* The layout view name.

Examples

- The following steps restore view FP in cell TOP in library myDesign from Session 1 in Session 2:

  Session 1

  1. Import the design
  2. After floorplanning, save the design in view FP.

     ```
     saveOaDesign myDesign TOP FP
     ```

  3. After placement, save the design in view PL.

     ```
     saveOaDesign myDesign TOP PL
     ```

  4. After routing, save the design.

```
saveOaDesign myDesign TOP RT
```

5. Exit the design.

```
exit
```

Session 2

➤ Restore the FP floorplan view from cell TOP in library myDesign.

```
restoreOaDesign myDesign TOP FP
```

saveTwf

`saveTwf` *`fileName`*

Saves the specified Timing Window Format (TWF) file.

You can use the `saveTwf` command after performing timing analysis.

Parameters

`fileName`	Specifies the name of the TWF file to save.

setDesignMode

```
setDesignMode
     [-help]
     [-reset]
     [-process integer]
```

Specifies the process technology value. You can use this command to change the process technology dependant default settings globally for each application instead of setting several mode options.

Use the `getDesignMode` command to return the current settings for the `setDesignMode` command.

When you specify a process technology value using the `setDesignMode` command, the Encounter software assigns coupling capacitance threshold values to the RC extraction filters automatically. These values determine whether the coupling capacitances of the nets in a design will be lumped to the ground.

Note: In detailed extraction mode, the grounding of coupling capacitances also depends on the capacitance filtering mode set by the `-capFilterMode` parameter of the `setExtractRCMode` command.

The following table lists the coupling capacitance threshold values that are applied based on the specified process node:

Process Node	Extraction Filters		
	coupling_c_th	**relative_c_th**	**total_c_th**
Above 130nm	3.0000	0.03	5.0000
130nm and below	0.4000	1	0.0000
90nm and below	0.2000	1	0.0000
65nm and below	0.1000	1	0.0000
45nm and below	0.1000	1	0.0000

The filtering values set by the Encounter software, which are based on the process node setting, attempt to capture the most significant effect of coupling capacitances on SI analysis, and are strongly recommended. However, you can choose to specify

different filtering thresholds using the `-coupling_c_th`, `-relative_c_th`, and `-total_c_th` parameters of the `setExtractRCMode` command.

In addition to the extraction filters, the software selects the extraction engine based on the process node. For 65nm and below process nodes, the software uses the improved detailed extraction engine, which provides scaling factors closer to 1 with better correlation to sign-off extraction. It also provides coupling capacitances to the wires on the layers above and below the victim wire.

Parameters

`-help`	Outputs a brief description that includes type and default information for the `setDesignMode` parameter. For a detailed description of the command and its parameters, use the `man` command: `man setDesignMode`.
`-reset`	Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setDesignMode` parameters to their default values.
`-process` *`integer`*	Specifies the process technology value to set for all the applications. Units in nanometers (nm). *Type:* Integer *Range:* 10-250 *Default:* 90

Examples

- The following command sets the specified process technology value to 80:

  ```
  setDesignMode -process 80
  ```

- The following command resets the `-process` parameter to its default value:

  ```
  setDesignMode -reset -process
  ```

- The following command also resets the `-process` parameter to its default value:

  ```
  setDesignMode -reset
  ```

Related Topics

- SI Closure in the *Encounter Flat Implementation Flow Guide*
- Partition SI Closure in the *Encounter Hierarchical Implementation Flow Guide*
- Top-Level SI Closure in the *Encounter Hierarchical Implementation Flow Guide*

setDoAssign

```
setDoAssign
     [on [-old] | off | stat]
     [-buffer buf_name]
     [-prefix prefix_name]
     [-excNetFile fileName]
```

Leaves the Verilog netlist unchanged during data import, and allows Encounter applications to replace Assign nets with buffers automatically.

Note: This command is not backward compatible for restoring placed and routed files. To save or restore the design, use the `defOut` or `defIn` commands.

The state of this command is saved in the configuration file. When you restore the design, this state is preserved, and you cannot change it. You can, however, use the following command to report the current state:

```
setDoAssign stat
```

Optionally, you can convert Assign nets to buffers. Assignments connecting two ports on a module from the top float to the top cell. The Assign net then connects the two ports on the top cell. To prevent this behavior you must insert a buffer in the module by using the following command:

```
setDoAssign -buffer buf_name on
```

If you use the `setDoAssign` command without parameters, Encounter returns the command status: `0` for `off`, and `1` for `on`.

You can use the `setDoAssign` command before you import the Verilog netlist.

Parameters

`-buffer` *`buf_name`*

Converts Assign nets to buffers. You must specify the type of buffer to use. This option changes the netlist.

`-old`

Uses the `setDoAssign on` command behavior from previous releases of Encounter. This removes Assign nets from the Verilog netlist unless the assign nets are connected to multiple primary ports. You must specify on when you use this parameter.

`on | off | stat`

Specifies whether to remove Assign nets from the Verilog netlist for a design.

Default: on

on	Leaves the netlist unchanged during data import, and allows Encounter applications to replace Assign nets with buffers automatically.
off	Does not remove Assign nets.
stat	Specifies whether the state of this command is on or off.

`-prefix` *prefix_name*

Specifies a prefix to use for the inserted buffer. When specified, the software creates buffer instances called *prefix_name_#*.

Note: You can use this parameter when you specify the `-buffer` parameter.

Default: The software creates buffer instances called `buf_dummy_#`

`-excNetFile` *fileName*

Does not insert buffers for the assign nets (analog nets) listed in the specified exclusion file. If the nets in the file appear in the LHS of an assignment, the software will not replace the assignment with a user-specified assign buffer.

Examples

- The following commands save the Verilog netlist, `myfile`, which contains buffers instead of Assign statements, and writes all empty modules to the file:

```
setDoAssign on -buffer buffer_name
setImportMode -keepEmptyModule true
saveNetlist myfile
```

Exclusion File Format

The exclusion file is a text file that contains a list of nets. If the nets in the file appear in the LHS of an assignment, the software will not replace the assignment with a user-specified assign buffer. Each line in the file begins with the keyword module. You can include comment

lines and empty lines, as long as they begin with #. Leading spaces, blank lines, and comments are ignored.

You can specify a module using the following format:

`module` *`moduleName netName`*

Where:

`moduleName` is the name of the module where the net is declared.

`netName` is the name of the net. The net name can be specified as a scalar, a bit-select, or a part-select. A bus name must be specified as a part-select, with the full range. It cannot be specified with the bus name only. When you specify the bus

Where:

`moduleName`	Specifies the name of the module where the net is declared
`netName`	Specifies the name of the net. The net name can be specified as a scalar, a bit-select, or a part-select. A bus name must be specified as a part-select, with the full range. It cannot be specified with the bus name only. When you specify the bus name as a part-select, there must be a space after the bus name, even if the name is unescaped.

The following example shows the format of an exclusion file:

```
# assign-LHS nets to be excluded from buffer insertion
#
module \SUB[1] \$-#/x2,2
module SUB2 y [0]
# For bus nets, such as [2:0] x, specify part-selects with full ranges.
module  SUB2 x   [2:0]
# If there is no module SUB3, the next line will trigger a warning.
module SUB3 y [1]
# y [1] can be expressed as y[1], for unescaped bus names.
module top y[1]
# If x[2] is not an assign-LHS net in module top, the next line has no effect.
module top x[2]
```

setExportMode

```
setExportMode
     [-help]
     [-reset]
     [-compress {true | false}]
     [-fullPinout {true | false}]
     [-implicitPortMapping {true | false}]
```

Controls certain aspects of how the software writes out a Verilog netlist.

Use the `getExportMode` command to return the current settings for the `setExportMode` command.

Parameters

`-fullPinout {true | false}`

Writes out all connected and unconnected pins in the netlist.

Default: `false`

`-help`

Outputs a brief description that includes type and default information for each `setExportMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setExportMode`.

`-implicitPortMapping {true | false}`

Writes out the netlist with implicit port mapping (that is, ordered port connections or position-based port connections) for hierarchical instances.

Default: `false`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setExportMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

Examples

Assume the following hierarchical module:

```
module
    block (out1, out2, out3, A, B, C);
    .
    .
    .
endmodule
```

- The following example shows how the `saveNetlist` command writes out an instance of the module `block` in the netlist by default:

  ```
  block i1(.out2(Y), .A(n1), .B(n2), .C(n3));
  ```

- The following command instructs the `saveNetlist` command to write out all connected and unconnected pins when it saves the netlist:

  ```
  setExportMode -fullPinout true
  ```

  The instance of module `block` is written in the saved netlist as follows:

  ```
  block i1(.out1(), .out2(Y), .out3(), .A(n1), .B(n2), .C(n3));
  ```

- The following command instructs the `saveNetlist` command to write out the netlist with implicit port mapping for hierarchical instances when it saves the netlist:

  ```
  setExportMode -implicitPortMapping true
  ```

  The instance of the module `block` is written in the saved netlist as follows:

  ```
  block i1( , Y, , n1, n2, n3);
  ```

- The following command resets the `-fullPinout` parameter to its default value:

  ```
  setExportMode -reset -fullPinout
  ```

- The following command resets all `setExportMode` parameters to their default values:

  ```
  setExportMode -reset
  ```

setImportMode

```
setImportMode
     [-help]
     [-reset]
     [-bufferTieAssign {true | false}]
     [-keepEmptyModule {true| false}]
     [-minDbuPerMicron dbuPerMicron]
     [-syncRelativePath {true | false}]
     [-timerMode {true | false}]
     [-treatUndefinedCellAsBbox {true | false}]
     [-useLefDef56 {true | false}]
     [-verticalRow {true | false}]
```

Controls aspects of how the software reads in the Verilog netlist and other design files.

Use the getImportMode command to return the current settings for the `setImportMode` command.

You can use the `setImportMode` command before loading a configuration file.

Parameters

`-bufferTieAssign {true | false}`

Determines how tie-high/tie-low Assign nets (assign *net* = 1'b1 or 1'b0) are handled in the output Verilog netlist.

If set to `true`, the software inserts a buffer instead of an Assign net in the output Verilog netlist. The buffer name must be specified with the setDoAssign command.

If set to `false`, the software inserts Assign nets in the output Verilog netlist.

Default: `false`

`-help`

Outputs a brief description that includes type and default information for each `setImportMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setImportMode`.

`-keepEmptyModule {true | false}`

Keeps empty modules.

If set to `false`, the software converts the empty modules into special leaf cells that do not have a physical library associated with them.

If the module is a hard macro, using a value of `true` or `false` produces the identical behavior: the software keeps the module.

Default: `true`

Note: Empty modules can be removed later from the design by using the `deleteEmptyModule` command.

`-minDbuPerMicron` *`dbuPerMicron`*

Sets the minimum database units per micron allowed when LEF files initialize the database units.

You can specify a value for the minimum database units per micron allowed when LEF files initialize the database units. This option is intended for users with LEF files that have database units that are not fine enough for NanoRoute to run properly – when NanoRoute requires the LEF database units to be finer than the manufacturing grid. If you do not want to modify your LEF file, you can use this option to override the LEF database units value.

The default value is `0`, which means that the software uses the LEF file units. The software uses the units set in the LEF file, if the LEF units are greater than the minimum database units you specify. Conversely, if the units you specify are greater than the LEF units, the software uses the units you specify. The value you set is saved so the next time you open the design it will have the same database units.

Default: `0`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setImportMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-synchRelativePath {true | false}`

Synchronizes all relative paths in the configuration file to the current working directory. You can load this configuration file from any directory without first changing your current working directory to the previous working directory where the configuration file was saved.

Default: `false` (You must change your working directory to the directory saved in the previous design session before you load the configuration file.)

`-timerMode {true | false}`

Note: For internal use only.

`-treatUndefinedCellAsBbox {true | false]`

Converts empty modules and cells that do not have physical cell definitions into blackboxes and partitions.

Empty modules do not contain instances or assignments, but could contain ports and net declarations.

Default: `false`

`-useLefDef56 {true | false}`

Reads in LEFand DEfFversion 5.6 or 5.7 files.

If set to `false`, the software reads in LEF and DEF files that are version 5.5 or older.

Default: `true`

`-verticalRow {true | false}`

Supports vertical rows.

Default: `false`

Examples

- The following commands save a Verilog netlist, `myfile`, that contains buffers instead of Assign statements, with all the empty modules written in the file.

```
setDoAssign on -buffer buffer1
setImportMode -keepEmptyModule true
saveNetlist myfile.v
```

- The following command synchronizes all relative paths in the configuration file to the current working directory:

```
setImportMode -syncRelativePath true
```

- The following command resets the `-syncRelativePath` parameter to its default value:

```
setImportMode -reset -syncRelativePath
```

- The following command resets all `setImportMode` parameters to their default values:

```
setImportMode -reset
```

setLayerExtId

```
setLayerExtId
     layerName
     ExtID value
```

Maps `ExtID` values to metal layers used in third-party tools.

You can use the `setLayerExtId` command at any time after you run the `loadConfig` or `restoreDesign` commands.

Parameters

`ExtID` *`value`*	Specifies the `ExtID` value.
`layerName`	Specifies the Encounter metal layer name to be mapped, for example: `M1`, `M2`, or `M3`.

setLibraryUnit

```
setLibraryUnit
     [-cap capUnit]
     [-time timeUnit]
```

Sets the time and capacitance unit values for the design, when the timing libraries contain differing values.

By default, the time and capacitance unit values for the design are set using the values specified in the timing libraries, when the libraries are loaded into the design. If the values in the timing libraries differ, the software generates warning messages and automatically sets the values to the default values. You can use the `setLibraryUnit` command to set different values, if you do not want the software to use the default values.

Note: You must specify the `setLibraryUnit` command before loading timing libraries into the design. The command does not have any effect, if the libraries are already loaded.

Parameters

`-cap` *`capUnit`*	Specifies the capacitance load unit in the timing library. *Default:* `1.0` pf
`-time` *`timeUnit`*	Specifies the time unit to be used for timing constraints. The valid values are 1ns, 1ps, 10ps, 100ps. *Default:* `1.0` ns

Example

- The following command sets the time unit for the design to `10`ps:

  ```
  setLibraryUnit -time 10ps
  ```

setNet

```
setNet
     {-net netName | -file fileName}
     -type {regular | special -setTermSpecial}
```

Marks the specified nets and their related pins as either special nets and pins, or regular nets and pins.

When the Verilog list is first loaded into the design, all LEF pins except those with `USE POWER` and `USE GROUND` are considered “regular pins” by default, and are routed by Trial Route and NanoRoute. You can use this command to mark nets (and their pins) as special prior to routing to keep NanoRoute from attempting to route them. This is useful when you have nets that need to be routed by hand, or with other specialized routers.

If you later decide you want the nets to be routed by NanoRoute, you can use the command to set them back to regular.

Note: The `setNet` command has no effect if the net is already routed. See the `convertNetToSNet` and `convertSNetToNet` commands if you want to change existing routes.

You can use the `setNet` command prior to routing with the NanoRoute router.

Parameters

`-file` *`fileName`*
: Changes all nets (and their related pins) in the specified file to be either special or regular. The file must contain a list of net names, with one net name per line.

`-net` *`netName`*
: Changes the specified net and its related pins to be either special or regular.

`-type {regular | special -setTermSpecial}`
: Specifies the type to which to change the net.

  `regular`
  : Changes the net and its pins to a regular net, so that NanoRoute will not route them.

  `special -setTermSpecial`

Changes the net and its pins to a special net to keep NanoRoute from attempting to route them.

Currently, you must specify `-setTermSpecial` for proper behavior.

Example

The following command sets the net `myAnalogNet` and its related pins as special:

```
setNet -net myAnalogNet -type special -setTermSpecial
```

The NanoRoute router ignores this net when routing.

setOasisOutMode

```
setOasisOutMode
     [-help]
     [-reset]
     [-cblockCompression {true | false}]
     [-removeNets {list_of_nets}]
     [-SEcompatible {true | false}]
     [-SEvianames {true | false}]
     [-snapToMGrid {true | false}]
     [-specifyViaName {default | format_string}]
     [-supportPathType4 {true | false}]
     [-textSize value]
     [-uniquifyCellNamesPrefix {true | false}]
     [-virtualConnection {true | false}]
```

Controls certain aspects of how the software generates OASIS files.

Use the getOasisOutMode command to return the current settings for the setOasisOutMode command.

You can use the setOasisOutMode command before loading a configuration file.

Parameters

-cblockCompression {true | false}

Reduces file size by using cblock compression. The oasisOut command can process merged OASIS files with cblocks, even if this parameter is set to false.
Default: true

-help

Outputs a brief description that includes type and default information for each setOasisOutMode parameter.

For a detailed description of the command and all of its parameters, use the man command: man setOasisOutMode.

-removeNets {*list_of_nets*}

Removes the specified nets from the output file.
Default: "" (empty string)

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setOasisOutMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-SEcompatible {true | false}`

Output instance only if `FOREIGN` is specified.
Default: `false`

`-SEvianames {true | false}`

Determines the naming convention used for vias declared in the LEF file.
Default: `false`

- Specify `true` to create unique names.
- Specify `false` to retain LEF via names.

`-snapToMGrid {true | false}`

Specifies whether to snap the nets' wires to the manufacturing grid.
Default: `false`

`-specifyViaName {default | format_string}`

Specifies the naming convention that the `oasisOut` command uses for via cells declared in the DEF file.
Default: `%t_VIA%u` (This default value is used so that the names of the via cells are compatible with versions 4.2 and earlier versions of the Encounter software.)

Note: Via names are truncated if they are longer than 32 characters. The software automatically appends a unique identifier to names if there is a name conflict.

You can use any of the following characters to specify a format string to generate names that give you information about the cell contents. Separate the characters with underscores (_) to improve readability. You can list them in any order.

- `%c`

  Specifies the number of columns.

- `%l(lcu)`

  Specifies the via layers. For the lower layer, use `%l` or `%l(l)`. For the cut layer, use `%l(c)`. For the upper layer, use `%l(u)`. You can specify one, two, or all three layers. Enclose the `l`, `c`, and `u` in parentheses, as in the following examples: `%l(c)`, `%l(lu)`.

- %n

  Specifies the number of cut rows and columns that make up a cut array.

- %r

  Specifies the number of cut rows.

- `%t`

  Specifies the top structure name.

- %u

  Specifies a unique number for the via.

- %v

  Specifies the via name in the DEF file.

For more information, see “Vias,” in the “DEF Syntax” chapter of the *LEF/DEF Language Reference*.

`-supportPathType4 {true | false}`

Converts `PATHTYPE4` to `PATHTYPE0`. If you specify `-supportPathType4 false`, the software converts `PATHTYPE4` paths to `PATHTYPE0` when outputting the path.

Default: `true`

`-textSize` *`value`*

Changes the size of the text used in the text labels in the OASIS file written by the Encounter software.
Default: `1`

- To increase the size of the text, specify a positive real number that is greater than 1, as in the following example:

  ```
  setStreamOutMode -textSize 2
  ```

- To decrease the size of the text specify a positive real number that is less than 1, as in the following example:

  ```
  setStreamOutMode -textSize 0.5
  ```

`-uniquifyCellNamesPrefix {true | false}`

Adds a prefix, instead of a suffix, to uniquified cell names. The prefix is the source filename. If this parameter is not specified, the software adds a suffix (also the source filename) to uniquified cell names.
Default: `false`

`-virtualConnection {true | false}`

Specifies whether a colon (`:`) label is appended to pin names for DEF pins with the `.extra`*`N`* syntax and to LEF pins with multiple ports (if `-outputMacros` is specified for LEF pin ports that have disjoint shapes).
Default: `ON`

- Specify `ON` to append a colon to pin names.
- Specify `OFF` not to append a colon to pin names.

Note: For more information, see the "DEF Syntax" chapter of the *LEF/DEF Language Reference*.

Examples

- The following command retains the via names defined in the LEF file:

  ```
  setOasisOutMode -SEvianames false
  ```

  When you type this command and run the `oasisOut` command, the Encounter software does the following:

  - For `block1`, the software creates `VIA12Default`, `VIA12DoubleCut`, and so on.
  - For `block2`, the software creates `VIA12Default`, `VIA12DoubleCut`, and so on.

If `block1` and `block2` are later merged into a top-level design, name collisions of the via masters occur. This result is acceptable if these are the same vias. However, if the vias are not the same, this command causes errors.

- The following command creates unique via names for vias defined in the LEF file:

  ```
  setOasisOutMode -SEvianames true
  ```

  When you type this command and run `oasisOut`, the Encounter software does the following:

  - For `block1`, the software creates `block1_VIA1`, `block1_VIA2`, and so on.
  - For `block2`, the software creates `block2_VIA1`, `block2_VIA2`, and so on.

  If `block1` and `block2` are later merged, there is no risk of name collision between via masters.

- The following command generates names for via cells. The names identify the top structure name (%v), lower via layer (`%l(l)`), and DEF via name (%v):

  ```
  setOasisOutMode -specifyViaName %t_%v_%l(l)
  ```

- The following command resets the `-textSize` parameter to its default value:

  ```
  setImportMode -reset -textSize
  ```

- The following command resets all `setOasisOutMode` parameters to their default values:

  ```
  setOasisOutMode -reset
  ```

setOaxMode

```
setOaxMode
     [-help]
     [-reset]
     [-cutRows {true | false}]
     [-dataModel value]
     [-fullLayerList {true | false}]
     [-fullPath {true | false}]
     [-instPlacedIfUnknown {true | false}]
     [-locking {true | false}]
     [-logicOnlyImport {true | false}]
     [-nativeLef {true | false}]
     [-pinPurpose {true | false}]
     [-reset]
     [-saveRelativePath {true | false}]
     [-updateMode {true | false}]
```

Controls aspects of how the Encounter software reads OpenAccess technology, and creates OpenAccess libraries.

Use the getOaxMode command to return the current settings for the setOaxMode command.

Parameters

`-cutRows {true | false}`

Cuts rows against placement obstructions, block halos, or power domain fence minimum gaps.

Default: `false`

`-dataModel value`

Specifies the highest version of the OpenAccess data model that can be written. You can specify a value of 3 or 4.

Lower data model databases can be created, if none of the newer data model features are required. For example, a data model 3 database can be created, even if `-dataModel 4` is specified.

Default: 4

`-fullLayerList {true | false}`

Imports the complete list of OpenAccess layers, when importing an OpenAccess design.

If you specify `false`, only OpenAccess layers defined with `oacMaterial` an type are imported.

Default: `true`

`-fullPath {true | false}`

Creates the `DEFINE` statements in the `lib.def` files using full paths. If you specify `-fullPath false`, the software creates the statements using relative paths.

Default: `false`

`-help`

Outputs a brief description that includes type and default information for each `setOaxMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setOaxMode`.

`-instPlacedIfUnknown {true | false}`

Maps the placement status of instances in the OpenAccess database to the placement status in the Encounter software. When set to `true`, the instances in the OpenAccess database that have an unknown placement status are treated as "placed" in the Encounter software. When set to `false`, instances with an unknown placement status are treated as "unplaced."

Default: `true`

`-locking {true | false}`

Locks the OpenAccess database when restoring the design. When set to `true`, other tools can still access the OpenAccess database in read-only mode, but cannot edit the database as long as it is locked for the Encounter session. To remove the lock, specify the `freeDesign` command, or exit the Encounter software.

Note: If you set the `-updateMode` parameter to `true`, the software automatically locks the database, even if the `-locking` parameter is set to `false`.

Default: `false`

The value of `-locking` cannot be changed while a design is in memory.

`-logicOnlyImport {true | false}`

Controls what information the software reads when importing an OpenAccess maskLayout database.

When set to `true`, the software only reads the logical (Verilog netlist equivalent) connectivity. When set to `false` (the default value), the software automatically reads physical information (floorplan, placement, routing, etc.) from the database, and updates physical net connectivity for power and ground connections.

Note: The `-logicOnlyImport` parameter only applies to OpenAccess maskLayout databases. For other types of OpenAccess databases, the software always only reads the logical connectivity.

Default: `false`

`-nativeLef {true | false}`

Directly reads OpenAccess technology (`LAYER`, `VIA`, `SITE`, etc.) and abstract (`MACRO`) information into the Encounter in-memory library data.

If you specify `false`, the software converts the OpenAccess technology and abstract information into LEF files first, then processes the LEF files to create the library data.

Default: `true`

`-pinPurpose {true | false}`

Writes out pin shapes with the purpose `pin`. When set to `false`, writes out pin shapes with the purpose `drawing`.

Default: `false`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setOaxMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-saveRelativePath {true | false}`

Defines the current working directory (`$cwd`) in the configuration file as "`.`". When set to `true`, a design can be restored, even if the directory path of the design file is changed.

Default: `true`

`-updateMode {true | false}`

Ensures that `restoreOaDesign` processes parameterized cells (PCells) and other custom objects correctly in the Encounter database.

Ensures that `saveOaDesign` updates the Encounter-modified portion of an OpenAccess database without making changes to PCells or other custom objects.

Default: `false`

The value of `-updateMode` cannot be changed while a design is in memory.

Examples

- The following command instructs the Encounter software to create the `DEFINE` statements using full paths:

  ```
  setOaxMode -fullPath true
  ```

- The following command cuts rows against placement obstructions, block halos, or power domain fence minimum gaps:

  ```
  setOaxMode -cutRows true
  ```

- The following command resets the `-fullPath` parameter to its default value:

  ```
  setOaxMode -reset -fullPath
  ```

- The following command resets all `setOaxMode` parameters to their default values:

  ```
  setOaxMode -reset
  ```

setStreamOutMode

```
setStreamOutMode
     [-help]
     [-reset]
     [-removeNets {list_of_nets}]
     [-SEcompatible {true | false}]
     [-SEvianames {true | false}]
     [-snapToMGrid {true | false}]
     [-specifyViaName {default | format_string}]
     [-supportPathType4 {true | false}]
     [-textSize value]
     [-uniquifyCellNamesPrefix {true | false}]
     [-version version_number]
     [-virtualConnection {true | false}]
```

Controls certain aspects of GDSII Stream files generated by the Encounter software.

Use the getStreamOutMode command to return current settings for the setStreamOutMode command.

Parameters

`-help`
: Outputs a brief description that includes type and default information for each `setStreamOutMode` parameter.

  For a detailed description of the command and all of its parameters, use the `man` command: `man setStreamOutMode`.

`-removeNets {`*`list_of_nets`*`}`
: Removes the specified nets from the output file.
  Default: `""` (empty string)

`-reset`
: Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setStreamOutMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-SEcompatible {true | false}`
: Output instance only if `FOREIGN` is specified.
  Default: `false`

`-SEvianames {true | false}`

Determines the naming convention used for vias declared in the LEF file.
Default: `false`

- Specify `true` to create unique names.
- Specify `false` to retain LEF via names.

`-snapToMGrid {true | false}`

Specifies whether to snap the nets' wires to the manufacturing grid.
Default: `false`

`-specifyViaName {default | format_string}`

Specifies the naming convention that the `streamOut` command uses for via cells declared in the DEF file.
Default: `%t_VIA%u` (This default value is used so that the names of the via cells are compatible with versions 4.2 and earlier versions of the Encounter software.)

Note: When the version of the GDS file is 3 (this is the default version), via names are truncated if they are longer than 32 characters. The software automatically appends a unique identifier to names if there is a name conflict.

You can use any of the following characters to specify a format string to generate names that give you information about the cell contents. Separate the characters with underscores (_) to improve readability. You can list them in any order.

- `%c`

  Specifies the number of columns.

- `%l(lcu)`

  Specifies the via layers. For the lower layer, use `%l` or `%l(l)`. For the cut layer, use `%l(c)`. For the upper layer, use `%l(u)`. You can specify one, two, or all three layers. You must enclose the `l`, `c`, and `u` in parentheses, as in the following examples: `%l(c)`, `%l(lu)`.

- %n

  Specifies the number of cut rows and columns that make up a cut array.

- %r

  Specifies the number of cut rows.

- `%t`

  Specifies the top structure name.

- %u

  Specifies a unique number for the via.

- %v

  Specifies the via name in the DEF file.

For more information, see “Vias,” in the “DEF Syntax” chapter of the *LEF/DEF Language Reference*.

`-supportPathType4 {true | false}`

Converts `PATHTYPE4` to `PATHTYPE0`. If you specify `-supportPathType4 false`, the software converts `PATHTYPE4` paths to `PATHTYPE0` when outputting the path.

Default: `true`

`-textSize` *`value`*

Changes the size of the text used in the text labels in the GDSII Stream file written by the Encounter software.
Default: `1`

- To increase the size of the text, specify a positive real number that is greater than 1, as in the following example:

  ```
  setStreamOutMode -textSize 10.0
  ```

- To decrease the size of the text specify a positive real number that is less than 1, as in the following example:

  ```
  setStreamOutMode -textSize 0.5
  ```

`-uniquifyCellNamesPrefix {true | false}`

Adds a prefix, instead of a suffix, to uniquified cell names. The prefix is the source filename. If this parameter is not specified, the software adds a suffix (also the source filename) to uniquified cell names.

`-version` *`version_number`*

Specifies the version for a created GDSII Stream file. If you use the `streamOut -merge` parameter, `streamOut` uses the highest version of the merged files, instead of the value specified by the parameter.
Default: 3

`-virtualConnection {true | false}`

Specifies whether a colon (`:`) label is appended to pin names for DEF pins with the `.extra`*`N`* syntax and to LEF pins with multiple ports (if `-outputMacros` is specified for LEF pin ports that have disjoint shapes).
Default: `true`

- Specify `true` to append a colon to pin names.
- Specify `false` not to append a colon to pin names.

Note: For more information, see the "DEF Syntax" chapter of the *LEF/DEF Language Reference*.

Examples

- The following command retains the via names defined in the LEF file:

  ```
  setStreamOutMode -SEvianames false
  ```

When you type this command and run the `streamOut` command, the Encounter software does the following:

- For `block1`, the software creates `VIA12Default`, `VIA12DoubleCut`, and so on.
- For `block2`, the software creates `VIA12Default`, `VIA12DoubleCut`, and so on.

If `block1` and `block2` are later merged into a top-level design, name collisions of the via masters occur. This result is acceptable if these are the same vias. However, if the vias are not the same, this command causes errors.

- The following command creates unique via names for vias defined in the LEF file:

  ```
  setStreamOutMode -SEvianames true
  ```

  When you type this command and run `streamOut`, the Encounter software does the following:

  - For `block1`, the software creates `block1_VIA1`, `block1_VIA2`, and so on.
  - For `block2`, the software creates `block2_VIA1`, `block2_VIA2`, and so on.

  If `block1` and `block2` are later merged, there is no risk of name collision between via masters.

- The following command generates names for via cells. The names identify the top structure name (%v), lower via layer (`%l(l)`), and DEF via name (%v):

  ```
  setStreamOutMode -specifyViaName %t_%v_%l(l)
  ```

- The following command resets the `-SEvianames` parameter to its default value:

  ```
  setStreamOutMode -reset -SEvianames
  ```

- The following command resets all `setStreamOutMode` parameters to their default values:

  ```
  setStreamOutMode -reset
  ```

Related Topics

- Importing and Exporting Designs chapter in the Encounter User Guide
  - "Converting the Database to GDSII Stream Format"

streamOut

```
streamOut
     gdsFile
     [-attachInstanceName attributeNumber]
     [-attachNetName attributeNumber]
     [-dieAreaAsBoundary]
     [-libName libraryName]
     [-mapFile mapFile]
     [-merge {listOfExternalGDSFiles}]
     [-uniquifyCellNames]
     [-mode {ALL | FILLONLY | NOFILL | NOINSTANCES}]
     [-offset x y]
     [-outputMacros]
     [-stripes numberOfStripes]
     [-structureName structureName | -noStructureName]
     [-units {100 | 200 | 1000 | 2000 | 10000 | 20000}]
```

Creates a GDSII Stream file of the current database. Writes a summary of errors and warnings to the Encounter log file.

- If you do not specify a map file, the Encounter software creates a generic map file, in GDSII Version 3 format. The generic file shows the mapping that was used, if none was specified. You must customize the file to make it appropriate for your design. You can change the version by using the `setStreamOutMode -version` parameter.
- You must specify all layers to stream out. Layers that are not specified in the map file are not included in the file created by this command.
- If you specify the `-merge` parameter and you want to merge files with a version number greater than 3, the software generates a file with the highest version number from the merged files.
- If you do not specify `-outputMacros`, `LEFPIN` and `LEFOBS` do not apply. If you do specify `-outputMacros` and `LEFPIN` and `LEFOBS` are not specified in the map file for the layers in the LEF macros, the GDSII structures for those macros will be empty.

Parameters

`-attachInstanceName` *`attributeNumber`*

Specifies the instance name value added to the attribute on SREF of the specified attribute number.

`-attachNetName` *`attributeNumber`*

Specifies that the net name will be added to the specified attribute number on the wiring shapes.

gdsFile

Specifies the GDSII Stream output file. Include the `.gz` extension to enable file compression (for example, `GDS_file.gds.gz`).

`-dieAreaAsBoundary`

Forces the `streamOut` command to write the die area as a polygon (that is, a boundary). Otherwise, the `streamOut` command writes the die area as a rectangle (that is, a box).

`-libName` *libraryName*

Specifies the library to convert to GDSII Stream format.
Default: `DesignLib`

`-mapFile` *mapFile*

Specifies the file used for layer mapping. This file is required for successful stream out, and is read by the next tool used in the design flow.

If you do not specify a file with this parameter, the Encounter software will read a `streamOut.map` file if one exists. If a `streamOut.map` file does not exist, the software creates a generic map file that you must customize for your design.

For information on the map file format, see GDSII or OASIS Map File Format in the *Encounter User Guide*.

`-merge {`*`listOfExternalGDSFiles`*`}`

Specifies a single file or list of files to merge. Checks all merged files for name collisions, and generates a warning if any names are changed. Uses the following conventions for new names:

- Replaces dot separators (`.`) with underscores (_). For example, `test.test_gds.gds` is renamed `test_test_gds_gds`.
- Adds unique identifiers. For example, if more than one cell is named `test.test.gds.gds`, they are renamed `test_test_gds_gds_1`, `test_test_gds_gds_2`, and so on.

All merge files must be GDSII Stream files. Compressed files are acceptable (include the `.gz` extension to enable file compression). The Encounter software automatically creates blackboxes when merging and writing macro files. It ignores any cells in the merge files that are not used in the design.

If you specify more than one file, separate the file names with spaces and enclose the list of file names in braces ({}) or double quotation marks (")

You can use a wildcard to read all `.gds` files in a specified directory. The usage is as follows:

```
streamOut -merge GDSIIdir/*.gds
```

where *`GDSIIdir`* is the name of the directory that contains the `.gds` files.

Use caution when using wildcards, as they might affect the order in which the software reads merge files.

`-mode {ALL | FILLONLY | NOFILL | NOINSTANCES}`

Identifies the layers to write.

Specify one of the following:

- `ALL`—Writes all layer information specified in `mapFile` as follows:
  - All instances
  - All via instances
  - All generated via cells
- `FILLONLY`—Writes only fill layers specified in `mapFile` as follows:
  - No instances
  - Only fill via instances
  - Only generated fill via cells
- `NOFILL`—Writes only non-fill layers specified in `mapFile`.
- `NOINSTANCES`—Writes wiring (including vias) only; does not write out `COMPONENT` instances.

`-noStructureName`

Enables you merge to files without loading an Encounter database when the `-merge` parameter is also specified.

`-offset` *`x y`*

Specifies, in microns, the location of the design in the GDSII output file. When the GDSII output file is saved, the design shifts to the specified location.

`-outputMacros`

Writes LEF abstract information such as LEF pin geometries and obstructions for macros. Writes one text label per port within a LEF macro pin plus one text label per shape for feedthrough pins.

Specify this parameter to create GDSII output that contains the LEF macro structures as well as the design data. You can also use this parameter to debug LEF macro definitions against verification DRC rules. The output contains macro pins and obstructions.

Note: `LEFPIN` and `LEFOBS` apply only when you specify this parameter. If you specify this parameter and `LEFPIN` and `LEFOBS` are not specified in the map file for the layers in the LEF macros the GDSII structures for those macros will be empty.

`-stripes` *`numberOfStripes`*

Specifies the number of stripes that divide the design.
Default: `1`

For example, setting the `-stripes` parameter to 2 divides the design into two equal windows, so the Encounter software creates three files. Setting the `-stripes` parameter to 3 divides the design into three equal windows, so the Encounter software creates four files. All polygons and shapes whose lower-left corner is in the current window are written with that stripe.

If the `-stripes` parameter value is greater than `1`, the naming conventions are as follows:

- `GDS_file.gds` (contains the top structure, structure reference to top structures of other stripes, and via definitions)
- `GDS_file.gds.1` (contains stripe number 1, `structureName.1`)
- `GDS_file.gds.2` (contains stripe number 2, `structureName.2`)

`-structureName` *`structureName`*

Specifies the superstructure that consists of all geometries that the map file writes out.
Default: Current design's top cell name

`-uniquifyCellNames`

Creates unique cell names if there are name collisions when the software merges files.

Creates unique cell names if there are name collisions when the Encounter software merges files.

Important

You must specify the `-merge` parameter if you specify this parameter. When you list the merge files, list the top-level file first, as the software uses the first name in the search path when renaming cells.

When a cell name is repeated, the software renames the cell *`cellName_filename`*, where *`filename`* is the name of the merge file in which the cell name appears.

The software changes the name of references to that cell in the merge file. The software changes the reference name if the referenced cell is defined in the same file, regardless of whether the cell is defined before or after the reference. If a reference in a merge file refers to a cell definition in a different merge file, then the reference is not changed in the GDSII Stream output file.
Default: If you do not specify this parameter, the software ignores duplicate cell names.

Note: You must specify the `-merge` parameter if you specify this parameter.

`-units {100 | 200 | 1000 | 2000 | 10000 | 20000}`

Specifies the resolution for values in the GDSII file. Choose `100`, `200`, `1000`, `2000`, `10000`, or `20000`.
Default: Units specified in LEF file

Examples

- The following commands create a GDSII Stream file:

```
streamOut amba_usb.gds -mapFile streamOut.map -libName DesignLib
  -attachInstanceName 12 -merge {ios.gds}
  -attachNetName 15
  -stripes 3 -units 2000 -mode ALL
  -dieAreaAsBoundary -outputMacros
```

- The following command creates blackboxes for cells not found in the merge libraries and writes macros only for those cells:

```
streamOut -merge {myFile1 myFile2 myFile3} -outputMacros
```

- The following command merges all files in `myDirectory` with the current database:

```
streamOut -merge myDirectory/*.gds
```

  If a merge file contains cells such as `A`, `B`, `C`, `D`, `X`, `Y`, and `Z`, and the design contains only cells `A`, `B`, and `D`, then the Encounter software merges only cells `A`, `B`, and `D`.

Related Topics

- Importing and Exporting Designs chapter in the *Encounter User Guide*
  - "Converting the Database to GDSII Stream Format"
  - "GDSII Map File Format"
- Signoff in the *Encounter Flat Implementation Flow Guide*

tdfIn

```
tdfIn
     [-isCaseInsensitive]
     [-r]
     [-withBackslash]
     [-scanChain]
     [-site siteName]
     fileName
```

Loads the individual Top Design Format (TDF) files. Because you cannot combine floorplanning data, placement data, and routing data into a single TDF file, you must be sure that each type of data is in a separate TDF file. Use one `tdfIn` command to load each type of data.

When you load a TDF file, you must be at the same level in the hierarchy where the TDF file was saved.

You can use the `tdfIn` command after importing the design. Before using the `tdfIn` or `tdfOut` commands, you must prepare layer mapping information in your LEF file. For more information, see "Importing and Exporting TDF Files", in the "Importing and Exporting Designs" chapter in the *Encounter User Guide*.

Parameters

fileName	Specifies the name of the TDF file to load.
-isCaseInsensitive	Specifies that the TDF file contains names that are case insensitive. Do not use this parameter if the names are case sensitive.
-r	Specifies that the TDF file contains routing data.
-scanChain	Specifies that the TDF file contains scan chain data.
-site *siteName*	Specifies a site name (defined in the LEF file) to be used to create rows. *Default:* The software attempts to find a site definition in the LEF file that matches the row size. If one cannot be found, the software generates a warning message and does not create the rows.
-withBackslash	Specifies that the TDF file contains back slashes in the names.

Example

- The following command loads the TDF placement file:

```
tdfIn TOPCHIP_SP.tdf
```

tdfOut

```
tdfOut
     [-blockOnly | -obstructOnly | -routingOnly | -stdCellOnly]
     [-doubleBackslash | -noBackslash]
     [-saveFPlanCmd]
     [-scanChain]
     fileName
```

Writes the Top Design Format (TDF) information to the specified file. Because you cannot combine floorplanning data, placement data, and routing data into a single TDF file, you must be sure that each type of data is in a separate TDF file. Use one `tdfOut` command to create a file for each type of data.

The `tdfOut` command does not apply to power strips.

You can use the `tdfOut` command after importing the design data. Before using the `tdfIn` or `tdfOut` commands, you must prepare layer mapping information in your LEF file. For more information, see "Importing and Exporting TDF Files" in the *Encounter User Guide*.

Parameters

`-blockOnly | -obstructOnly | -routingOnly | -stdCellOnly`

Determines which kind of TDF placement or routing data is written out to the specified file. Use only one parameter per `tdfOut` command.

- `-blockOnly` Writes out only the block placement data.
- `-obstructOnly` Writes out only the standard cell placement obstruction layer data.
- `-routingOnly` Writes out only the routing data.
- `-stdCellOnly` Writes out only the standard cell placement data.

`-doubleBackslash` Ensures that names with backslash characters have double backslashes.

`-noBackslash` Removes the backslash from a name.

Default: Names are written out with a single backslash.

`-saveFPlanCmd` Writes out the floorplan command replay file for the Apollo tool. The filename is `*.tdf.fp.cmd`.

`-scanChain`	Includes scan chain information in the output file.

Examples

- The following command writes only block placement data to `mytdffile`:

```
tdfOut -blockOnly mytdf
```

uniquifyNetlist

```
uniquifyNetlist
     { -top topCellName
          [-v | -q]
          [-suffix suffixName]
          [-uniquifyDummyModule]
          [-exclude excludedCellFile | -select selectedCellFile]
          uniquifiedNetlistFile inputNetlistFile ...
     | -conf configFile
          [-v | -q]
          [-suffix suffixName]
          [-uniquifyDummyModule]
          [-exclude excludedCellFile | -select selectedCellFile]
          uniquifiedNetlistFile
     }
```

Note: This is a UNIX prompt command. This program is located in the `~/bin` directory.

Uniquifies the design's netlist and writes the uniquified design to a netlist file.

When you *uniquify* a netlist, you create a unique module definition for every module in the netlist. (The `uniquifyNetlist` command does *not* create unique definitions for leaf cells in the design.) You can load your uniquified netlist by specifying the revised netlist name in the Design Import form, or in your configuration file.

Note: By default, the software checks and reports whether the netlist is unique after flattening the top cell during design import. If the netlist is not unique, you must uniquify it outside of the Encounter software using the the `uniquifyNetlist` command. You can instruct the software to automatically uniquify the netlist directly after flattening the top cell when the netlist is not unique by adding the following line to your configuration file:

```
set rda_Input(ui_uniquify_netlist) {1}
```

Important

Because the `uniquifyNetlist` command expands the module definitions in the netlist file, the netlist file may be significantly larger after you use this command.

You can use this UNIX-prompt command whenever you have made netlist changes—for example, after IPO, ECO, clock tree synthesis, or scan chain insertion.

Parameters

`-conf` *`configFile`*	Specifies the name of a configuration file that contains information about which netlist is to be uniquified.

`-exclude` *excludedCellFile*

Specifies a file that contains a list of cell names to be excluded from the uniquified netlist. This parameter is mutually exclusive with the `-select` parameter.

inputNetlistFile — One or more Verilog netlist files to be uniquified. Separate input netlist files with a space. You can specify the files in any order.

Note: You can use this parameter *only* with the `-top` parameter—*not* with the `-conf` parameter.

`-select` *selectedCellFile*

Specifies a file that contains a list of cell names to be uniquified. This parameter is mutually exclusive with the `-exclude` parameter.

`-suffix` *suffixName* — Specifies a suffix to indicate uniquified cells.

`-top` *topCellName* — Specifies the design name to be uniquified. Use this parameter when a netlist contains more than one design.
Default: If you do not specify this parameter, the `uniquifyNetlist` command assumes that the input netlist contains a single design.

`-uniquifyDummyModule`

Uniquifies any empty modules with `assign` statements (“dummy modules”).

Note: If you want to uniquify only one empty module (“dummy module”), you can use the `-select` parameter to specify such a module for uniquifying.

uniquifiedNetlistFile

The netlist file name that the uniquified design is written to.

`-v | -q` — Specifies whether progress messages appear in the Encounter console:

- `-v` — Allows progress messages to appear in the Encounter console (“verbose”).
- `-q` — Suppresses progress messages in the Encounter console (“quiet”).

Example

Uniqifies the entire design, `workingDesign.v`, and writes an uniquified netlist, `uniqueDesign.v`:

```
uniquifyNetlist uniqueDesign.v workingDesign.v
```

unlockOaDesign

```
unlockOaDesign
     [-help]
     lib
     cell
     view
```

Unlocks the specified OpenAccess design.

A design cannot be opened if `setOaxMode -locking` is set to `true`, and the design is currently locked. You can use the `unlockOaDesign` command to free the lock so that other tools can access and edit the database. You also can use the command to override the lock, if the design is incorrectly locked.

Parameters

cell	Specifies the name of the top cell to unlock.
-help	Outputs a brief description that includes type and default information for each `unlockOaDesign` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man unlockOaDesign`.
lib	Specifies the name of the OpenAccess library in which the design was saved.
view	Specifies the name of the cell view to unlock.

Example

- The following command unlocks the view `layout` in the cell `dtmf_chip` in the library `designLib`:

  ```
  unlockOaDesign designLib dtmf_chip layout
  ```

13

Interactive ECO Commands

addHierInst

```
addHierInst
     -cell cellName
     -hinst hinstName
```

Adds a hierarchical module to the design.

You can use this command after importing a design.

Parameters

`-cell` *`cellName`*

Specifies the master of the module. Encounter creates a new, empty master if the cell name is new.

`-hinst` *`hinstName`*

Specifies the name of the module instance to add.

Examples

- The following command adds a hierarchical instance `A/B/spare` of module type `spareModule` to the design:

  ```
  addHierInst -cell spareModule -hinst A/B/spare
  ```

addInst

```
addInst
     [-moduleBased verilogModule]
     [-physical]
     -cell cellName
     -inst instName
     [-loc llx lly [-ori orient]]
```

Adds an instance and places it in the design.

When `-inst` is specified, the new instance is bound with the correct cell in the power domain.

When `-inst`, and `-loc` are specified, the location also gives a power domain, which is checked against the power domain of the instance. If the power domains are different, a warning is issued.

Parameters

`-cell` *cellName*

Specifies the master of the instance.

`-inst` *instName*

Specifies the name of the instance to add and place.

`-loc` *llx lly*

Specifies the location of the placed instance.
Default: If you do not specify a location, Encounter places the instance at the design origin.

`-moduleBased` *verilogModule*

Adds the new instance to the specified verilog module.

`-ori` *orient*

Specifies orientation of the placed instance.
Default: If you do not specify an orientation, Encounter uses `R0`.

`-physical` Places a physical instance without updating the netlist.

Examples

- The following command adds buffer instance i1/i2 at location 100, 200:

```
addInst -cell BUF1 -inst i1/i2 -loc 100 200
```

- The following command adds buffer instance `insta` to the verilog module `HIER_2`:

```
addInst -moduleBased HIER_2 -cell BUFX2_6 -inst insta
```

addModulePort

```
addModulePort
     moduleName | -
     [-moduleBased verilogModule]
     portName {input | output | bidi}
     [-bus n1:n2]
```

Adds a port or a bussed port to a module.

Parameters

`-bus n1:n2`

Adds a bussed port to the module. Specify the bus range (the beginning and end of the bus). Use integers to specify the range.

`-moduleBased verilogModule`

Adds the new port to the specified verilog module.

`moduleName | -`

Specifies the hinst to which you want to attach the port. To specify the top module, enter `'-'`.

`portName`

Specifies the name of the port to be added.

The specified `portName` can be a new port name or any of the existing net names to which you want to add the port.

The new port name can be the same as the existing net name. This port is attached directly to the existing net name if you specify the port direction (scalar port).

`input | output | bidi`

Specifies whether the port is input, output, or bidirectional.

Examples

- The following command creates an input port `p1` on instance `i1/i2/i3`. The port `p1` must be a new port name in instance `i1/i2/i3`. Instance `i1/i2/i3` contains no nets name `p1`:

  ```
  addModulePort i1/i2/i3 p1 input
  ```

- The following set of commands adds a hierarchical block and then adds a bussed port to the block.

```
addHierInst -cell i_block1 -hinst i_block1
addModulePort i_block data output -bus 31:0
```

- The following command adds an `output` port, `testPort` to the verilog module `hier_3`:

```
addModulePort -moduleBased hier_3 testPort output
```

addNet

```
addNet
     [-moduleBased verilogModule]
     netName
     [-physical]
     [-bus startID:endID]
```

Adds a net to the design. The net can be logical or physical.

Parameters

`-bus` *`startID:endID`*

Creates a bussed Verilog net. The *`startID`* and *`endID`* indicate the first and last bits on the bus. You must separate the start and end IDs with a colon (:).

`-moduleBased` *`verilogModule`*

Adds the new net to the specified verilog module.

`netName` Specifies the net to add.

Note: You must specify *`netName`* before you specify other parameters. The software issues an error message if you do not specify the *`netName`* first.

`-physical` Adds a physical net.

Examples

- The following command adds net `i1/n1`:

  ```
  addNet i1/n1
  ```

- The following command adds a bussed net `i1/n1` with bus bits 0 to 7:

  ```
  addNet i1/n1 -bus 0:7
  ```

attachDiode

```
attachDiode
     [-prefix prefix]
     -diodeCell diodeCellName
     -pin instName termName
     [-loc x y
     [-orient orient]]
```

Adds an antenna diode to a post-routed design. Encounter places the diode as close as possible to the specified pin.

When *instName* is specified, the new instance is bound with the correct cell in the power domain.

When `-loc` is specified, the location also gives a power domain, which is checked against the power domain of the pin. If the power domains are different, a warning is issued.

Parameters

`-diodeCell` *diodeCellName*

Specifies the name of the diode cell master

`-loc` *x y* Specifies the location of the lower left corner of the diode.

`-orient` *orient*

Specifies orientation of the placed diode. To specify this parameter, you must also specify `-loc`.

`-pin` *instName termName*

Specifies the pin associated with the diode.

`-prefix` *prefix* Specifies the prefix name of the diode cell instances. Encounter appends integers to differentiate the instances.

Example

- The following example adds a diode of type `D1` for `term2` on instance `i2`, at location `500`, `600`:

```
attachDiode -diodeCell D1 -pin i2 term2 -loc 500 600
```

attachIOBuffer

```
attachIOBuffer
     [-baseName baseName]
     [-excludeClockNet]
     {-in cellName | -out cellName}
     [-markFixed]
     [-selNetFile selNetFileName]
     [-excNetFile excNetFileName]
     [-port]
     [-suffix suffixName]
```

Adds buffers to the I/O pins of a block and places the buffers near the I/O pins. When a top-level input (output) is connected to a power domain other than the default, the added buffer is matched to the hierarchy of the destination (source) cell and placed inside the power domain.

This command is power-domain aware. To take advantage of this feature when the netlist contains level shifters, load the shifter table before running this command.

Through the `-port` and `-suffix` parameters, you can control the instance and net names created when you use `attachIOBuffer`.

You can use this command after loading a design. Use the `-markFixed` parameter before you run `placeDesign`.

Parameters

`-baseName` *`baseName`*

Specifies the base name of the new nets. For nets with the same base name, you should consider the hierarchical index and add `1`. For example, if there are several nets with an `abc` base name, ending on `abc98`, then the first new net with the same base name is `abc99`.

`-excludeClockNet`

Does not connect the buffer on clock nets. You must trigger the clock net markings prior to running `attachIOBuffer`, with the `timeDesign` command. To query whether a given clock net is marked, use `dbIsNetClock` *`netName`*.

`-excNetFile` *`excNetFileName`*

Specifies the file that contains the names of nets to exclude from the buffer attachment operation.

`-in` *`cellName`* | `-out` *`cellName`*

Specifies the cell name of the input (`-in`) or output (`-out`) buffer.

`-markFixed` Marks newly-inserted buffers as `Fixed`.

`-port` Prepends the port name to the name of the net or instance created.

`-selNetFile` *`selNetFileName`*

Specifies the file that contains the names of nets to include in the buffer attachment operation.

`-suffix` *`suffxName`* Appends a string to name of the net or instance created.

Example

- The following command attaches buffers `bufx4` to the I/O pins, and marks the buffers as `Fixed`. The new nets have a basename of `abc`:

  ```
  attachIOBuffer -baseName abc -in bufx4 -markFixed
  ```

- The following command prepends the port name and appends the `_buf` string to the name of the instance and net created by `attachIOBuffer`:

  ```
  attachIOBuffer -port -suffix "_buf"
  ```

  Resulting instance name:

  ```
  port1_I_buf
  ```

  Resulting net name:

  ```
  port1_N_buf
  ```

attachModulePort

```
attachModulePort
     {moduleName | -}
     portName
     netName
```

Attaches a port in the specified instance (or top level) to a net.

Parameters

moduleName | -	Specifies the module to which you want to attach the port. To specify the top module, enter '-'.
netName	Specifies the net to which you want to create to attach to the port.
portName	Specifies the port you want to create on the module.

Examples

- The following commands create a port `p1` on instance `i1/i2/i3` and a net `i1/i2/n1`. The `attachModulePort` command then connects the created port `p1` on instance `i1/i2/i3` to the net `i1/i2/n1`:

```
addModulePort i1/i2/i3 p1
addNet i1/i2/n1
attachModulePort i1/i2/i3 p1 i1/i2/n1
```

- The following command attaches port `in` on the top module to `net123`:

```
attachModulePort - in net123
```

attachTerm

```
attachTerm
     [-moduleBased verilogModule]
     [-noNewPort]
     instName
     termName
     netName
     [-port portName | -pin refInstName refPinName]
```

Attaches a terminal to a net. If the terminal already connects to a different net, the software first detaches the terminal from the current net, then attaches it to the new net.

Note: Detaching a terminal that drives an output terminal of a module produces a Verilog violation at the output terminal if you use `detachTerm`. Instead, use `attachTerm` to attach the terminal to a new net. The `attachTerm` command automatically detaches the terminal from the net connecting to the output terminal, then attaches the terminal to the net you specify.

Parameters

`instName`
Specifies the instance containing the terminal. If the instance name does not exist, the software displays an error message and the command stops.

`-moduleBased` *`verilogModule`*
Attaches the terminal to the specified verilog module.

`netName`
Specifies the name of the net to attach to the terminal. If the net name does not exist, Encounter displays an error message and the command stops.

`-noNewPort`
Prohibits Encounter from creating hierarchical ports when it attaches a terminal. If the terminal cannot connect to the net through existing ports, Encounter displays an error message and the command stops.
Default: If you do not specify this parameter, Encounter creates hierarchical ports as needed.

`-pin` *`refInstName refPinName`*
Specifies the pin *`refPinName`* on instance *`refInstName`* connected to the net that Encounter connects to the terminal. You cannot specify the `-port` parameter if you use this parameter.

`-port` *`portName`*

Specifies the hierarchical port used to connect the terminal with the net. If you specify this parameter, the hierarchical port must exist in the module that contains the instance. The hierarchical port must connect to the net. This parameter lets you use a specific port to maintain the same netlist topology, which simplifies equivalence checking later in the design flow. You cannot specify the `-pin` parameter if you use this parameter.
Default: If you do not specify this parameter, Encounter uses existing ports or creates new hierarchical ports as necessary to connect the terminal to the net.

`termName`

Specifies the name of the terminal that Encounter connects to the specified net. If the terminal name does not exist, Encounter displays an error message and the command stops.

Examples

- The following command attaches terminal `in1` of instance `i1/i2/i3` to net `i1/i2/net26`:

  ```
  attachTerm i1/i2/i3 in1 i1/i2/net26
  ```

  If `i1/i2/i3` does not exist, or if `in1` is not a terminal of `i1/i2/i3`, or if `i1/i2/net26` does not exist, the software displays an error message and the command stops.

- The following command attaches terminal `in1` of instance `i1/i2/i3` to net `net27`, using hierarchical port `myPort`:

  ```
  attachTerm i1/i2/i3 in2 net27 -port myPort
  ```

  The `myPort` port must exist in the module definition for `i1/i2`, and `myPort` must already connect to `net27`. If this is not the case, the software displays an error message and the command stops.

- The following command attaches terminal `in3` on instance `i1/i2/i3` through existing ports to `net28`:

  ```
  attachTerm -noNewPorts i1/i2/i3 in3 net28
  ```

  If ports do not exist, the software displays an error message and the command stops.

- The following command attaches terminal `Y` of instance `testInst` to the verilog module `hier_t3` using the port `testPort`:

  ```
  attachTerm -moduleBased hier_t3 testInst Y testPort
  ```

deleteEmptyModule

`deleteEmptyModule`

Deletes modules and their instantiations if the following criteria are met:

- The netlist is unique
- The module does not contain any instances, or the module contains only unconnected instances of other empty modules
- The instances of the module are not connected to other instances or ports

The `deleteEmptyModule` command does not delete used empty modules, where there exists at least one connected instance of the empty cell. For example, black boxes typically are empty, but they are still connected to other instances or ports.

Note: A design can contain empty modules for different reasons. Modules can become empty due to optimization processes. A design also can contain MCPO modules that were read in with the netlist. To convert empty modules into special leaf cells when reading in the netlist, see the `setImportMode` command.

deleteInst

```
deleteInst
     instanceName
     [-moduleBased verilogModule]
```

Deletes an instance from a design. Encounter checks whether the instance is physical or logical, and deletes the instance accordingly.

You can use wildcards (*?) to specify the instances you want the tool to delete.

Parameters

instanceName Specifies the name of the instance to delete.

-moduleBased *verilogModule*

Deletes the instance from the specified verilog module.

Example

- The following command deletes instance i1/i2:

  ```
  deleteInst i1/i2
  ```

- The following command deletes all instances whose names begin with i:

  ```
  deleteInst i1/i*
  ```

- The following command deletes the instance insta from the verilog module HIER_2:

  ```
  deleteInst -moduleBased HIER_2 insta
  ```

- The following command deletes all instances in the SPGM module:

  ```
  deleteInst * -moduleBased SPGM
  ```

deleteModulePort

```
deleteModulePort
     [-moduleBased verilogModule]
     portName -
```

Disconnects the specified portname from its net and deletes the port.

You can use wildcards (*?) to specify the nets you want Encounter to delete.

Parameters

`-`	Specifies the top-level module.
`-moduleBased` *`verilogModule`*	Specifies the name of the verilog module.
`portName`	Specifies the name of the port to be deleted

Example

- The following command deletes `port1` from the top-level module:

```
deleteModulePort - port1
```

deleteNet

```
deleteNet
     netName
     [-moduleBased verilogModule]
```

Deletes a net from a design. Encounter checks whether the net is physical or logical, and deletes the net accordingly.

You can use wildcards (*?) to specify the nets you want Encounter to delete.

Parameters

`-moduleBased` *verilogModule*
: Deletes the net from the specified verilog module.

netName
: Specifies the name of the net to delete.

Example

- The following command deletes `net1`:

  ```
  deleteNet net1
  ```

- The following command deletes all nets whose name begin with `net1`:

  ```
  deleteNet net1*
  ```

detachModulePort

```
detachModulePort
     moduleName
     portName
```

Detaches the net connected to the specified port on the specified instance.

Parameters

moduleName	Specifies the module from which you want to detach the net.
portName	Specifies the port on the module.

Example

- The following command detaches port p1 from moduleA:

```
detachModulePort moduleA p1
```

detachTerm

```
detachTerm
     [-moduleBased verilogModule]
     instName
     termName
     [netName]
```

Disconnects a terminal from a net.

Note: Detaching a terminal that drives an output terminal of a module produces a Verilog violation at the output terminal if you use `detachTerm`. Instead, use `attachTerm` to attach the terminal to a new net. The `attachTerm` command automatically detaches the terminal from the net connecting to the output terminal, then attaches the terminal to the net you specify.

Parameters

instName	Specifies the instance that contains the terminal you want to detach.
-moduleBased *verilogModule*	Detaches the terminal from the verilog module.
netName	Specifies the net that is already connected to the terminal. If the terminal does not connect to the specified net, or if the net does not exist, the software displays an error message and the command stops. The software uses this parameter for error checking only.
termName	Specifies the terminal to disconnect. If the terminal does not exist on the specified instance, the software displays an error message and the command stops.

Example

- The following command disconnects terminal `in1` of instance `i1/i2/i3`:

  ```
  detachTerm i1/i2/i3 in1 i1/i2/net26
  ```

  If instance `i1/i2/i3` does not exist, or terminal `in1` is not a terminal of `i1/i2/i3`, the software displays an error message and the command stops. The net `i1/i2/net26` is specified, so if net `i1/i2/net26` does not exist, or if the terminal is not already connected to net `i1/i2/net26`, the software displays an error message and the command stops.

displayBufTree

```
displayBufTree
     netName
     [-outfile fileName]
```

Displays a buffer tree associated with the specified net.

Parameters

netName	Specifies the name of the net for which you want to display the buffer tree.
-outfile *fileName*	Specifies the text file containing buffer locations.

Example

- The following command highlights instances and wires in the buffer tree associated with the net `reset` in the design display area:

  ```
  displayBufTree reset
  ```

ecoAddRepeater

```
ecoAddRepeater
     {-net netName | -term inst1/term inst2/ ...}
     -cell masterCellName
     [-loc {x y | x1 y1 x2 y2} [-radius µm]] |
     [-relativeDistToSink value [-radius µm]] |
     [-offLoadSlack ns [-radius µm]] |
     [-noPlace] |
     -offLoadAtLoc {x y}
     [-evaluateOnly | -evaluateAll]
```

Adds either a single buffer or two inverters on a net. This allows you to optionally specify the location and radius, the location relative to the sink pin, and the buffer insertion based on the slack. This command also checks for and observes power domains based on the specified instance name and location. This command cuts wires by default.

This command does not modify `dont_touch` nets if `setEcoMode -honorDontTouch` is set to `true`.

Note: The `ecoAddRepeater` command is simultaneous setup/hold and MMMC aware. To make the command simultaneous setup/hold aware, set the global variable `set_global timing_enable_simultaneous_setup_hold_mode` to `true`. The simultaneous setup/hold mode can only be used for timing analysis. It cannot be used for any physical implementation steps.

You can use this command to add a single inverter if you set `setEcoMode -LEQCheck` to `false`. When set to `true`, it can add a pair of inverters.

Parameters

`-cell` *`masterCellName`*

Specifies the master name for the cell to be inserted. Either a buffer or an inverter can be specified as the master cell. If an inverter is specified, two inverters in series will be added. The cell should have timing information in the library.

`-evaluateOnly`

Evaluates the effect on timing if you add the new cell.

The values are not applied in the database.

`-evaluateAll`

Evaluates the effect on timing for all the cell types available for the new cell. The timing report shows the effects of all the cell types, enabling you to select the best cell for your design.

The values are not applied in the database.

`-loc {`*`x y | x1 y1 x2 y2`*`}`

Specifies the x and y coordinates for the location of the buffer or inverter pair to insert. If no location is specified, the tool will automatically determine a location.

If the cell is a buffer and location is specified, the newly created buffer is placed at the specified location. If the cell is an inverter pair and only one location is specified, both inverters will be placed at the same location.

`-net` *`netName`* Specifies the name of the net where the buffer will be inserted.

`-noPlace` Specifies that the inserted cells should not be placed. Only the logical change in connectivity will be made.

`-offLoadAtLoc {x y}` Inserts a buffer at a the specified location on a net to reduce the load on the net driver. The tool determines the buffer direction by tracing backward on the net to find the buffer driving the wire segment at the new buffer location. The driven segment is connected to the input terminal of the new buffer.

`-offLoadSlack` *`ns`* Inserts a buffer to offload the paths with less than the specified slack threshold from the driver of a net with multiple cells. This option ignores the *`term`* option, if it is provided.

`-radius` *`um`* Specifies the radius in which the added instances are free to move. If no legal location can be found in the specified radius, the cells would be placed at the specified location resulting in an overlap with other cells. In that case, you should perform legalization.

`-relativeDistToSink` *`value`*

Specifies the location of the buffer based on its distance from the sink or the driver pin. The value is a number between 0 and 1. A low value (0.1) places the buffer near the sink; a high value (0.9) places the buffer near the driver. The fraction is based on the length of the wire.

This option works when one term is provided; it does not work if no term or multiple terms are specified.

`-term` *`inst1`*… Specifies a list of terms with one common net. If -term is not specified, all the terms of the net will be buffered.

Note: You can specify the following in the `setEcoMode` command:

```
setEcoMode -honorFixedStatus false #to modify FIXED instance.
```

```
setEcoMode -honorDontUse false #to modify instances which have dontUse
attribute on cells.
setEcoMode -honorDontTouch false #to modify instance/nets which are marked
dontUse.
```

Examples

- The following command adds a buffer near the receiver:

  ```
  ecoAddRepeater -net wire_I13 -relativeDistToSink 0.1 -cell BUFX4
  ```

- The following command adds a buffer near the driver:

  ```
  ecoAddRepeater -net wire_I13 -relativeDistToSink 0.9 -cell BUFX4
  ```

- The following command adds a buffer in a specified location

  ```
  ecoAddRepeater -net wire_I10 -loc 51.5 85.67 -cell BUFFX4
  ```

- The following command adds an inverter by specifying the terminal list:

  ```
  ecoAddRepeater -net n1 -cell BUFX4 -offLoadSlack -0.1
  ```

- The following command adds an inverter by specifying the slack time:

  ```
  ecoAddRepeater -net wire1_B2 -cell INVX4 -offLoadSlack -0.1
  ```

- The following command inserts a buffer in a specified net and evaluates timing after you added the buffer:

  ```
  ecoAddRepeater -net ibias1 -cell BUFFX2 -evaluateOnly
  ```

- The following command inserts a buffer in a specified net and evaluates the effect on timing for all the buffer types available for the added buffer:

  ```
  ecoAddRepeater -net ibias1 -cell BUFFX2 -evaluateAll
  ```

ecoChangeCell

```
ecoChangeCell
     -inst instNames
     -upsize | -downsize | -cell masterCellName
     [-pinMap oldpin1 newpin1 oldpin1 newpin1 ...]
     [-evaluateOnly | -evaluateAll]
```

Upsizes or downsizes the specified instance based on the available footprint. Cell mapping is also possible as long as the output pins are functionally the same. This command also checks for and observes power domains.

This command does not modify `dont_touch` nets if `setEcoMode -honorDontTouch` is set to `true`.

- You cTo correct hold violations, use the following commands:

  ```
  optDesign -postRoute -hold
  optDesign -postRoute -hold -si
  ```

  The first command repairs hold violations. The second command performs the following operations:

  - Analyzes cross coupling capacitance effects on glitch and noise
  - Back-annotates delays due to cross coupling capacitance
  - Reruns timing analysis
  - Repairs hold violations

- To input a SPEF file for use during postroute signal integrity optimization, use the following commands:

  ```
  setSIMode -usePrevSpefFile spefFileName
  optDesign -postRoute -si
  ```

  Note: The SPEF file you specify is used in the first loop of SI fixing only. From the second loop onwards, the software re-extracts the design.

- To run postroute setup fixing based on SDF in non-MMMC mode, use the following commands:

  ```
  setOptMode -setupSdfFile SETUP.sdf
  optDesign -postRoute -useSDF
  ```

- To run postroute hold fixing based on SDF in non-MMMC mode, use the following commands:

  ```
  setOptMode -setupSdfFile SETUP.sdf -holdSdfFile HOLD.sdf
  optDesign -postRoute -hold -useSDF
  ```

  Note: You must specify a setup SDF file when doing hold fixing.

You can use this command to change a buffer to a single inverter if you have set `setEcoMode -LEQCheck` to `false`. When set to `true`, it can change a buffer to a pair of inverters.

Note: If you use the `ecoChangeCell` command to modify a '`fixed`' instance, the software issues a warning message.

Instead, you can specify the following using the setEcoMode command:

- `setEcoMode -honorFixedStatus false`, to modify FIXED instance.
- `setEcoMode -honorDontUse false` to modify instances which have dontUse attribute on cells.
- `setEcoMode -honorDontTouch false` to modify instance/nets which are marked dontUse.

Note: The `ecoChangeCell` command is simultaneous setup/hold and MMMC aware. To make the command simultaneous setup/hold aware, set the global variable `set_global timing_enable_simultaneous_setup_hold_mode` to `true`. The simultaneous setup/hold mode can only be used for timing analysis. It cannot be used for any physical implementation steps.

Parameters

`-evaluateAll`	Evaluates the effect on timing for all the cell types available for the new cell. The timing report shows the effects of all the cell types, enabling you to select the best cell for your design. The values are not applied in the database.
`-evaluateOnly`	Evaluates the effect on timing, if you modify the specified cell. The values are not applied in the database.
`-inst` *`instNames`*	Specifies the names of the instances to be changed.
`-upsize`	Upsizes the specified instance with the next available cell in the same footprint unless the specified cell is already of the highest driving strength. Upsizing an instance that drives a large load can improve the driver delay and the transition time at the receivers.
`-downsize`	Downsizes the specified instance with the next available cell in the same footprint unless the specified cell is already of the lowest driving strength. Downsizing an instance on the noncritical path can reduce the loading of its driver on the critical path.
`-cell` *`masterCellName`*	Specifies the name of a specific cell to be substituted in the design. The function of the original cell has to match the function of the new cell for all the outputs used by the old cell. The only exception is if the original cell is a buffer and the new cell is an inverter. In that case, two inverters in series will be swapped with the original buffer. The number of input pins have to be exactly the same for both the original master cell and the new master cell.

`-pinMap` *`oldpin1 newpin1 oldpin1 newpin2`*

Specifies the pin mapping for the new cell based on the old cell. This option is required if the new master cell has different pin names than the original cell. The net connected to *`oldpin1`* of the original cell should be connected to *`newpin1`* of the new cell, and so forth.

Examples

- Upsizes the specified instance:

  ```
  ecoChangeCell -inst Top_I18 -upsize
  ```

- Downsizes the specified instance:

  ```
  ecoChangeCell -inst Top_I18 -downsize
  ```

- Modifies the specified instance according to the specified cell:

  ```
  ecoChangeCell -inst Top_I18 -cell BUFX20
  ```

- Changes a specified buffer to an inverter pair:

  ```
  ecoChangeCell -inst Top_I18 -cell BUFX20
  ```

- Changes to another cell with a different pin name:

  ```
  ecoChangeCell -inst Top_I27 -cell TXOR2X2 -PINMAP A X B Y Y Z
  ```

- Downsizes the specified instance and evaluates timing after you modified the instance:

  ```
  ecoChangeCell -inst Top_I18 -downsize -evaluateOnly
  ```

ecoCompareNetlist

```
ecoCompareNetlist
     { -def referenceFileName | [-inMemory | -external] }
     [-logical]
     -outFile fileName
```

Compares a specified netlist with the design in the database for structural or logical equivalence, and creates a file containing the differences found.

Parameters

`-def` *`referenceFileName`*

Specifies the name of the DEF file that Encounter uses to compare with the design.

`-external` Compares the post-ECO netlist to an external netlist.

`-inMemory` Compares the post-ECO netlist to the netlist in memory. This is the default behavior.

`-logical` Compares the netlists for logical equivalence.

`-outFile` *`fileName`*

Specifies the file that contains the differences between the specified DEF file and the design in the database.

Example

- The following command compares the `myFile.def` file against the design, and writes the results to `differences.file`:

  ```
  ecoCompareNetlist -def myFile.def -outfile differences.file
  ```

ecoDefIn

```
ecoDefIn
     [-postMask [-suffix suffixName]]
     [-useGACells GACoreSite]
     [-reportFile fileName]
     [-cleanFEData]
     [-ecoDef {new_DEF_fileName}]
     filename
```

Restores physical information from an old design and compares this information with the design in memory.

You must first perform the following steps:

1. `loadCofig`
2. Change the netlist.
3. `commitConfig`

This command uses the specified DEF file in the following ways for pre-mask and post-mask ECO:

- Used in pre-mask mode (without the `-postMask` parameter) this command behaves as follows:
  - New cells and nets

    Cells and nets that exist in memory but not in the DEF file become unplaced.
  - Deleted cells or nets

    Cells and nets that exist in the DEF file but not in memory are discarded.
  - Modified nets

    Nets found both in the DEF file and in memory, but whose connections are different, are marked as “modified” for further processing during `ecoRoute`.
  - Matched cells/nets

    Nets and cells that exist both in the DEF file and in memory, and have the same connections, are placed as described in the DEF file. This includes the soft matching capability used in `defIn`.

    Soft matching happens when a DEF net name does not have an equivalent name and another net is found in memory, that has the same connections as described in the DEF.

The most common case for soft matching is when nets have multiple aliases in a hierarchical design “net1” = “inst1/net2” = “inst1/inst2/net3” and so on. Any of these net names can be used in the DEF and can have the same connections.

Without soft matching, net ‘a’, for example, is removed and net ‘b’ is created in it’s place, resulting in ripping of the wire.

- In post-mask mode (with the -postMask parameter), the software can only change nets, not cells. This command behaves as follows:
  - New added cells and nets

    Cells and nets that exist in memory but not in the DEF file become unplaced. Later, these cells but will be mapped to spare cells when you use ecoPlace -useSpareCells.
  - Deleted cells

    Cells and instances that exist in DEF but not in memory are added to a spare cell list in Encounter: They are not discarded. Deleted cells are renamed with their original name and the suffix you specify. The default suffix is _SPARE. For example, cellA is renamed cellA_SPARE. The software retains deleted nets and their routing, which ecoRoute reroutes later.
  - Deleted nets

    Nets that exist in DEF but not in memory are kept and processed later when you use the ecoRoute command.
  - Modified nets

    Same behavior as in the pre-mask mode.
  - Matched cells and nets

    Same behavior as in the pre-mask mode.
  - Physical cells

    Cells that exist in the DEF file but not in memory, which are marked with +SOURCE DIST in the DEF file, are restored.

If you specify -useGACells, this implies a post-mask mode. This option behaves as follows:

- For cells that match the specified GA core site:
  - New added GA cell that exists in memory but not in the DEF file are left unplaced.
  - New GA cells that exist in the DEF file but not in memory are deleted.

- For cells that do not match the specified GA core site, the software treats them as standard cells as described in `-postMask` mode.

If you specify a report filename, this command reports the following information:

- New cells and nets found in memory but not in the DEF file
- Deleted cells and nets found in DEF but not in memory
- Nets whose connection does not match
- Cells and nets that match in Encounter and DEF

Parameters

`-cleanFEData` Specifies that the routed data is DRC clean. ECO routing will skip routing analysis.

Do not use this parameter if your data is from `trialRoute`.

`-ecoDef {`*`new_def_fileName`*`}`

Specifies the name of a partial DEF file. The tool compares the original routed DEF file or Encounter database with this file and marks the changed instances, changed location of the existing instances, and changed segments or wires.

`fileName` Specifies the name of the routed/original DEF file to load.

`-postMask` Runs this command in post-route mode.

`-reportFile` *`fileName`*

Specifies the name for the report generated by the `ecoDefIn` command.
Default: `ecoDefIn.rpt`

`-suffix` *`suffixName`*

Appends the specified suffix to cells that appear in the DEF file but have been deleted in the new netlist.
Default: _SPARE

Examples

The following example reads a DEF file, `myOld.def`, for post-mask ECO.

```
ecoDefIn -postMask -suffix _SPARE -reportFile ecoDefIn.rpt myOld.def
```

When the software converts a deleted instance to a spare cell, it adds the _SPARE suffix to the instance. This command generates a report `ecoDefIn.rpt`.

The following example shows a report file:

```
DELINST DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649 INVX1
(converted to spare cell
DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649_SPARE)
ADDINST DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649 BUFX8
MODIFYNET dtmf_chip_TIELO_Y_r2c1
ADDNET DTMF_INST/UNCONNECTED_000
ADDNET DTMF_INST/UNCONNECTED_001
ADDNET DTMF_INST/TDSP_CORE_INST/UNCONNECTED_000
ADDNET DTMF_INST/TDSP_CORE_INST/UNCONNECTED_009
ADDNET DTMF_INST/TDSP_CORE_INST/UNCONNECTED_010
```

ecoDeleteRepeater

```
ecoDeleteRepeater
     -inst inst1 [inst2]
     [-evaluateOnly]
```

Deletes a buffer or pair of inverters connected back-to-back. This command merges wires after ECO.

This command does not modify `dont_touch` nets if `setEcoMode -honorDontTouch` is set to `true`.

Note: `The ecoDeleteRepeater` command is simultaneous setup/hold and MMMC aware. To make the command simultaneous setup/hold aware, set the global variable `set_global timing_enable_simultaneous_setup_hold_mode` to `true`. The simultaneous setup/hold mode can only be used for timing analysis. It cannot be used for any physical implementation steps.

Parameters

`-evaluateOnly`	Evaluates the effect on timing, if you delete the specified cell. The values are not applied in the database.
`-inst inst1 [inst2]`	Specifies a buffer or inverter to delete. If you specify an inverter, the command finds another inverter on the net and deletes the back-to-back pair in order to preserve the logical equivalence. Instances are deleted only if they are tied to each other. To specify a pair of instances, enclose the instances with braces. `-inst {inst1 inst2}`

Note: You can specify the following using the setEcoMode command:

- `setEcoMode -honorFixedStatus false`, to modify FIXED instance.
- `setEcoMode -honorDontUse false` to modify instances which have dontUse attribute on cells.
- `setEcoMode -honorDontTouch false` to modify instance/nets which are marked dontUse.

Examples

- The following command deletes the specified buffer, `A/inst1`:

  ```
  ecoDeleteRepeater -inst A/inst1
  ```

- The following command deletes a back-to-back inverter pair containing `A/inv1`:

  ```
  ecoDeleteRepeater -inst A/inv1
  ```

- The following command deletes the instance containing `DTMF_INST/TDSP_CORE_INST/i_10177` and evaluates the effect on timing:

  ```
  ecoDeleteRepeater -inst DTMF_INST/TDSP_CORE_INST/i_10177 -evaluateOnly
  ```

- The following command deletes instances `DTMF_INST/RESULTS_CONV_INST/FE_PHC845_r1644_2_` and `DTMF_INST/RESULTS_CONV_INST/FE_PHC845_r1643_3_`:

  ```
  ecoDeleteRepeater -evaluate_only -inst {DTMF_INST/RESULTS_CONV_INST/
  FE_PHC845_r1644_2_ DTMF_INST/RESULTS_CONV_INST/FE_PHC845_r1643_3_}
  ```

ecoDesign

```
ecoDesign
     [-postMask]
     [-tieCell "cellName1 cellName2"]
     [-addHierPortsForTieOff]
     [-modifyOnlyLayers MLb:MLt]
     {-spareCells spareCellName}
     [-suffix suffix]
     [-useGACells GACoresite]
     [-noEcoPlace]
     [-noEcoRoute]
     [-ecoSDC newSDCFile]
     [-def fileName]
     [-fillerPrefix prefix]
     design.enc.dat top_cell new_netlist
```

Takes an Encounter database and a modified netlist as input and performs ECO operations. It restores the design, examines the changes in the new netlist, and automatically implements the required changes with `ecoPlace` and `ecoRoute`.

Parameters

`-addHierPortsForTieOff`

Allows the added tie cells to connect to the tie pins across hierarchical boundaries if other constraints, such as maximum fanout and maximum distance, allow it.

This parameter can be used only with `-postMask` and `-tieCell` options.

`-def` *`fileName`*

Specifies the DEF file you want the software to load automatically when performing ECO. This parameter automatically assigns the location of the new ECO cells. As a result, you do not need to run `ecoPlace` after performing the ECO.

`design.enc.dat top_cell new_netlist`

Specifies the name of the design, top cell, and the new netlist.

`-ecoSDC` *`newSDCFile`*

Specifies the name of the new sdc file.

`-fillerPrefix` *`prefix`*

Specifies the filler cell instance prefix. The `ecoDesign` command removes all filler cells before placing the ECOs. If you do not specify a prefix, `deleteFiller` removes filler instances based on the `setFillerMode` command settings. If you have not set `setFillerMode`, then no filler cells are removed.

`-modifyOnlyLayers MLb:MLt`

Specifies the layers to be modified during the ECO process.

`-noEcoPlace` Prevents placement during the ECO process.

`-noEcoRoute` Prevents routing during the ECO process.

`-postMask` Specifies that the changes are being made on a post-mask design.

`-spareCells` *`spareCellName`*

Specifies the name of the spare cell to use during the ECO process.

`-suffix` *`suffix`*

Specifies the suffix to be appended to the spare cells. You can specify this parameter only if you specify `-postMask.`

Default: _SPARE

`-tieCell “`*`cellName1 cellName2`*`”`

Specifies the existing tie cells to be reused to make tie connections for newly created SPARE instance in the design. You can only specify a maximum of two tie-cells, where one cell must be a tie-high driver, and the other a tie-low driver. The two tie-cell names must be surrounded by quotation marks.

This parameter can be used only with `-postMask` option.

`-useGACells` *`GA_site`*

Specifies the site name of the GA cell. Check the site name defined in the LEF file.

Examples

The following command performs a pre-mask ECO:

```
ecoDesign original.enc.dat myDesign myDesign.new.v
```

The following command and options are used to implement a post-mask ECO:

```
ecoDesign -postMask -modifyOnlyLayers 2:3 -spareCells *spare* original.enc.dat
myDesign myDesign.new.v
```

ecoOaDesign

```
ecoOaDesign
     lib
     cell
     view
     -ecoVerilogFile input_fileName
     [-reportFile output_fileName]
```

Imports an OpenAccess database and an ECO-modified netlist. Restores the physical objects in the database, examines the changes in the new netlist, and implements the required connectivity changes in the logical database without making changes to the physical objects.

Parameters

cell	Specifies the OpenAccess cell name of the top cell to import.
-ecoVerilogFile *input_fileName*	Specifies a Verilog netlist. The netlist is written out from the specified *lib cell view* and must be hand-modified to incorporate the required ECO changes.
lib	Specifies the OpenAccess library name.
-reportFile *output_fileName*	Specifies a report file that shows the additions and removals of instances and nets in the new Verilog netlist. *Default:* eco.log
view	Specifies OpenAccess view name to import.

ecoPlace

```
ecoPlace
     [-help]
     [-useSpareCells |
     -useGACells GACoreSite |
     -useGAFillerCells GAFillerCells]
     [-fixPlacedInsts]
```

Incrementally places the unplaced standard cells. In a pre-mask flow, this command moves the unplaced standard cells into the core area. In a post-mask flow (if you specify `-useSpareCells` or `-useGACells`), the software can map the unplaced cells to the available spare cells.

You can use this command after the `restorePlace` or `ecoDefIn` command when some newly added cells are still unplaced.

Parameters

`-fixPlacedInsts`
: Specifies that instances with the PLACED status are not moved.

`-help`
: Outputs a brief description that includes type and default information for each `ecoPlace` parameter.

  For a detailed description of the command and all of its parameters, use the `man` command: `man ecoPlace`.

`-useGACells` *`GACoreSite`*
: Maps only cells that match the GACoreSite to available spare cells. Chooses a spare cell and changes the connections from the unplaced cell to the selected spare cell. You can use this parameter only for post-mask ECO.

  Note: You cannot specify the `-useGAFillerCells` parameter if you use this parameter.

`-useGAFillerCells` *`GAFillerCells`*

Specifies a list of Gate Array (GA)-style filler cells that the tool can replace with GA cells during post-mask ECO. You can only use this parameter in a post-mask ECO flow. The new logic instances adding during a post-mask flow must already be present in the netlist.

Note: You cannot specify the `-useGACells` parameter if you use this parameter.

`-useSpareCells`

Maps unplaced cells to spare cells during placement. First, checks whether all unplaced cells match spare cells. Next, chooses a spare cell and changes the connections from the unplaced cell to the selected spare cell. Finally, deletes the unplaced cell. You can use this parameter only for post-mask ECO.

This option observes the power domain when selecting the spare cell and maps an unplaced cell to the spare cell in the power domain, based on the instance name of the cell. For example, if module A/B is defined for power domain PD2, this option will map an unplaced cell A/B/inst1 to a spare cell in module A/B.

ecoRemap

```
ecoRemap
     [-remapSuffix string]
     [-rpt fileName]
     [-inst instName]
     [-spareSuffix suffixName]
     [-allowConstantTie]
```

Maps unplaced newly-added cells to spare cells. All newly-added ECO cells are unplaced. No new cells are added or deleted physically, so you can use this command as a post-mask operation or when you want to preserve diffusion layers.

When you use this command, you do not know which spare cells are available, and whether an ECO modification could create timing or DRV problems. The software automatically analyzes the functionality of the newly-added cells and remaps them to available spare cells. The software analyzes the logic and performs changes to improve timing and minimize DRVs.

This command is power-domain aware. During remapping, the spare cells must be selected from the same power domain.

Before you use this command, you must have spare cells in your design. Cadence recommends using simple gates. The more spare cells you add, the more possibilities the command can explore. Also, more spare cells mean that the command can more easily improve timing by finding a spare cell nearby.

Note: Adding spare cells can introduce leakage power, so you must consider the trade-offs between leakage power and the number of spare cells.

To add spare cells, use the following commands:

- `createSpareModule`
- `deleteSpareModule`
- `placeSpareModule`

This command has the following limitations:

- The `ecoRemap` command cannot tie unused pins on complex gates, which can cause a mapping failure.
- The `ecoRemap` command can work only on logic cones. There must be at least one two-input gate, no multi-output gates, no sequential gates, and no multi-sink nets.
- The `ecoRemap` command can work only on unplaced cells.

Parameters

`-allowConstantTie`

Maps a cell to a spare cell containing more inputs than the cell in order to improve timing. The extra inputs are tied to a constant.

`-inst` *`instName`*

Specifies a placed cell instance to map to a nearby spare cell. This could result in better timing or DRV fixing. After the placed cell is remapped, the instance becomes an spare cell. Specify an suffix for the spare cell by using `-spareSuffix`.

`-remapSuffix` *`string`*

Appends specified suffix to the instance names of the remapped cells.
Default: `_remap`*`num`*, where *`num`* is an automatically incremented number which ensures that the suffixes are unique.

When the software remaps an unplaced cell to multiple cells, the software appends the specified suffix to the instance name of the remapped cells.

For example, if the software maps `NANDX3` with instance name `I1` to `ANDX4` and `INVX8`, then the `ANDX4` instance name is `I1_remap1` and the `INVX8` instance name is `I1_remap2`.

When multiple unplaced cells are mapped to a spare cell, the new instance of the spare cell is *`instance1_instance2`*`_remap`.

For example, if unplaced cells `ANDX4` (instance `I1`) and `INVX8` (instance `12`) are remapped to `NANDX3` (`spare1`), the new instance of the `NANDX3` spare cell is `I1_I2_remap`.

`-rpt` *`fileName`*

Specifies the output file that contains the mapping information.
Default: `ecoRemap.rpt`

`-spareSuffix` *`suffixName`*

Specifies a suffix for a cell specified by `-inst` after the cell becomes a spare cell.

Default: _SPARE

Examples

The following command remaps newly-added cells to spare cells, appends suffix `myCell`, and generates report `myReport.rpt`:

```
ecoRemap -remapSuffix myCell -rpt myReport.rpt
```

ecoRoute

```
ecoRoute
     [-modifyOnlyLayers bottomLayer:topLayer]
     -handlePartition
```

This command is part of an ECO flow process and is based on the NanoRoute® router. Based on information from the ecoDefIn command, this command preprocesses the design for the NanoRoute router and launches ECO routing within the router. This command retrieves the router's mode settings and returns them to their original state at the end of the ECO routing operation.

When you specify two layers to modify, and the second layer is a higher layer number (for example, `-modifyOnlyLayers 2:3`), `ecoRoute` checks for pin accessibility for new or modified nets and drops `VIA12` as necessary.

When a net is modified, some parts of the wire must be reassigned to the new nets. Initially, a whole wire between two cells is assigned to `netA`. When a buffer is inserted on `netA`, `ecoRoute` cuts the wire segment and assigns the left side of the wire to `netA` and the remaining segment on the right to `netB`.

When you specify the `-handlePartition` option, the command performs hierarchical ECO routing.

You must first run `ecoDefIn` and `ecoPlace` before you use this command.

You can run ECO routing in multiple-CPU processing mode. For information on the commands to use, see the Multiple-CPU Processing Commands chapter.

Parameters

`-handlePartition`	Specifying this option enables the command to perform hierarchical ECO routing. The hierarchical ecoRoute performs routing by maintaining 'partition-awareness' (retaining the partition pin location and number of pins of the partitions) for ECO nets. **Note:** This parameter is applied only for hierarchichal designs that might need to be able to perform partitioning. This parameter is used for ART-based post-route optimization flow. For more information, see "Using ART in Hierarchical Designs" chapter in the *Encounter User Guide*.

`-modifyOnlyLayers` *`bottomLayer:topLayer`*

Enables the NanoRoute router to remove segments of net routing in the specified layer range, from `bottomLayer` to topLayer, inclusively. You must include a colon (`:`) between the `bottomLayer` and `topLayer` values. This option also restricts eco routing to carry out only on eco nets.

Default: ECO routing can occur on any layer and any nets.

Note: You might want to use `-modifyOnlyLayers` to specify all layers in order for `ecoRoute` to avoid touching non-eco nets. In some cases, approach could be faster and result in fewer DRCs.

This command is similar to the `-routeWithEco`, `-routeSelectedNetOnly`, and `-routeEcoOnlyInLayers` parameters for the `setNanoRouteMode`. For more information on the `ecoRoute` command, see the ECO Flows chapter of the *Encounter User Guide*.

Examples

The following example allows the NanoRoute router to perform ECO routing only on or between layers 2 and 3:

```
ecoRoute -modifyOnlyLayers 2:3
```

If DRCs exist after you run the `ecoRoute` command, this could be because `ecoRoute` cannot touch non-eco nets. You can reroutie the DRC area as shown below:

```
setNanoRouteMode -quiet routeWithEco true
setNanoRouteMode -quiet
routeEcoOnlyInLayers 1:3
setNaonoRouteMode -quiet routeSelectdNetOnly false
globalDetailRoute 100.0 1200.0 350.0 600.0
```

Related Topics in the User Guide

- "ECO Flows" chapter of the *Encounter User Guide*
- "Accelerating the Design Process by Using Multiple-CPU Processing" chapter of the *Encounter User Guide*

Related Topic in the Menu Reference

- Multiple CPU Processing form in the "Design Menu" chapter of the *Encounter Menu Reference*

ecoSwapSpareCell

```
ecoSwapSpareCell
     [-suffix string]
     [-preservePin {list_of_spare_cell_pins}]
     [-keepScan]
     instname
     spareCellInstName
```

Swaps a placed cell with a cell in the spare cell list.

You can only specify this command after you have run `ecoPlace` and `specifySpareGate`.

This command can swap cells of the same cell type (same footprint and identical pin name). Cells of different sizes can also be swapped. The spare cell is connected according to the specified global net connections.

If the cell to be swapped is unplaced, it is mapped to the spare cell. *`instName`* is deleted, and its connection is transferred to the spare cell. If the cell to be swapped is placed, it is swapped with the spare cell and is renamed to *`instNameSuffix`* if the `-suffix` option is used. If a suffix is not specified, the *`instName`* cell is renamed to *`spareCellInstName`*. The *`instName`* cell's connections are transferred to *`spareCellInstName`*. The input of *`instName`* is `tielo`, based on the global connection definition.

Parameters

`instName`	Specifies the name of the cell to swap.
`-keepScan`	Keeps the scan connection for both the spare cell and the specified instances. Before you can use this parameter, you must first identify the scan connections by running a scan trace.
`-preservePin {list_of_spare_cell_pins}`	Specifies a list of pins whose connection you want to preserve while swapping spare cells. The same pin on both the spare cell and specified instances is preserved. If both the instance and the spare cell are placed instances, the `-preservePin` parameter is effective for both cells. If the instance is unplaced, `-preservePin` only affects the spare cell because the instance is deleted.
`spareCellInstName`	Specifies the spare cell in the spare cell list to swap with the placed cell.
`-suffix` *`string`*	Specifies a string that is appended to the *`instName`* after swapping.

Examples

The following example swaps a placed cell, `DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649`, with spare cell `spare1`:

```
specifySpareGate -inst *spare*
ecoPlace -useSpareCells
checkPlace
ecoSwapSpareCell DTMF_INST/TDSP_CORE_INST/PROG_BUS_MACH_INST/i_9649 spare1
```

getEcoMode

```
getEcoMode
     [-honorDontTouch]
     [-inheritNetAttr]
     [-LEQCheck]
     [-prefixName]
     [-refinePlace]
     [-spreadInverter]
     [-updateTiming]
     [-honorDontUse]
     [-honorFixedStatus]
     [-quiet]
```

This command displays the following information about the parameters for the corresponding `setEcoMode` command in the Encounter log file and in the Encounter console: name, possible values, type (Boolean, string, and so on), and source of current setting (default, user).

Parameters

`parameter_names`	Returns information for the specified parameters. You can specify one or more parameters. **Note:** `-inheritNetAttr` is for Cadence internal use only.
`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setEcoMode` parameters in Tcl format.

Examples

- The following command returns the current setting for the `-LEQCheck` parameter:

  ```
  getEcoMode -LEQCheck
  ```

  The software returns the following information:

  ```
  -LEQCheck true                    # bool, default=true
  true
  ```

- The following command returns the parameters for all the `setEcoMode` command:

  ```
  getEcoMode
  ```

  The software returns the following information:

```
-honorDontTouch true                # bool, default=true
-honorDontUse true                # bool, default=true
-honorFixedStatus true                # bool, default=true
-inheritNetAttr true # bool, default=true
-LEQCheck true                    # bool, default=true
-prefixName {ECO}                 # string, default=ECO
-refinePlace true                 # bool, default=true
-spreadInverter true              # bool, default=true
-updateTiming true                # bool, default=true
{honorDontTouch true} {-honorDontUse true} {-honorFixedStatus true}
{inheritNetAttr true} {LEQCheck true} {prefixName {ECO}} {refinePlace true}
{spreadInverter true} {updateTiming true}
```

- The following command returns the current setting for the `-spreadInverter` parameter in Tcl list format only:

  ```
  getEcoMode -spreadInverter -quiet
  ```

  The software returns the following information:

  ```
  true
  ```

- The following command returns the current settings for all `setEcoMode` parameters in Tcl list format only:

  ```
  getEcoMode -quiet
  ```

  The software returns the following information:

  ```
  {honorDontTouch true} {-honorDontUse true} {-honorFixedStatus true}
  {inheritNetAttr true} {LEQCheck true} {prefixName {ECO}} {refinePlace true}
  {spreadInverter true} {updateTiming true}
  ```

loadECO

```
loadECO
     [-postMask | -useGACells GACoreSite} [-suffix]]
     fileName …
     [-postMask]
```

Reads a file containing ECO directives and applies the changes to the current netlist.

You can use this command at any time after you import the design.

This command also performs consistency checks while the ECO commands are applied and after the changes are complete. It also checks for and observes power domains. This command produces results compatible with `ecoDefIn`, so you will use `ecoPlace` and `ecoRoute` to complete the ECO process.

The ECO directives do not affect routing, except in the following cases:

- `DELETENET` deletes routing attached to a specified deleted net.
- `DELETEINST` deletes an instance but leaves the wires attached to the instance dangling. You can then repair dangling wires with the router after ECO is complete.

If you want to reroute the entire design, delete routing before running the `loadECO` command.

This command supports ECOs for gate array designs through the `-useGACells` and `-postMask` parameters.

If an error occurs when the `loadECO` command reads the ECO file, the `loadECO` command stops without reading more commands from the file, and writes error messages to the `encounter.log` file.

Note: The first line of the ECO file you read must be the following:

```
FORMATVERSION 2
```

For information on the commands you can include in the ECO command file, see “ECO Directives,” in the *Encounter User Guide*.

Parameters

fileName	Specifies the name of the file that contains the ECO directives. You can specify more than one file. File names must be separated by one space.
`-suffix`	Specifies the suffix this command adds to spare cells created by deleting instances. The default suffix is SPARE.

`-useGACells` *`GACoreSite`*

Uses gate array cells when adding or deleting instances. This parameter applies to gate array designs only. If `ADDINST` is specified in the ECO file, the default behavior for `ADDINST` applies. If `DELETEINST` is included in the ECO file, the software deletes an instance if the instance master uses the same site as the specified *`GACoreSite`*, which can be the name of any gate array site defined in the LEF file. If you do not specify this parameter, `DELETEINST` removes the instance from the netlist.

`-postMask`

When adding an instance, the new cell will be unplaced. The unplaced cell will map to the spare cell during placement with the `ecoPlace -useSpareCells` command.

When the software deletes an instance, it renames the instance to an instance name at the original level of hierarchy, and appends the specified suffix. If you do not specify the `-suffix` parameter, this parameter appends SPARE.

Example

- The following command reads `myfile` and runs the ECO commands it contains:

  ```
  loadECO myfile
  ```

- The `loadECO` command issues a summary report on completion, for example:

```
***************** Load ECO Summary *****************
Number of New Nets:                          1028
Number of New Instances:                     833
Number of Changed Connections:               248
Number of Changed Instances:                 950
Number of Deleted Nets:                      0
Number of Deleted Instances:                 0
```

For an example of an ECO file, see “Example ECO File” in the *Encounter User Guide*.

setEcoMode

```
setEcoMode
     [-help]
     [-reset]
     [-honorDontTouch {true|false}]
     [-honorDontUse {true|false}]
     [-honorFixedStatus {true|false}]
     [-inheritNetAttr {true | false}]
     [-LEQCheck {true|false}]
     [-prefixName prefix]
     [-refinePlace {true}false}]
     [-spreadInverter {true|false}]
     [-updateTiming {true|false}]
```

Controls certain behaviors of ECO commands.

Use the getEcoMode command to return the current settings for the `setEcoMode` command.

Parameters

`-help`
: Outputs a brief description that includes type and default information for each `setEcoMode` parameter.

  For a detailed description of the command and all of its parameters, use the `man` command: `man setEcoMode`.

`-honorDontTouch {true|false}`
: Checks for the don't touch attribute on nets and instances. If you specify a terminal list, this parameter checks each terminal for the don't touch attribute. The `ecoAddRepeater` command honors the don't touch nets. The `ecoDeleteRepeater` and `ecoChangeCell` commands honor don't touch instances.

  Default: `true`

`-honorDontUse {true|false}`
: Checks for the don't use attribute on cells. The `ecoAddRepeater`, `ecoDeleteRepeater`, and `ecoChangeCell` commands honor don't use cells.

  Default: `true`

`-honorFixedStatus {true|false}`
: Does not allow preplaced, fixed instances to be resized.

  Default: `true`

`-inheritNetAttr {true | false}`

Inherits the net attribute.

Default: `true`

`-LEQCheck {true|false}`

Enables functionality checking when swapping cells. The software can swap cells that are logically equivalent.

Default: `true`

`-prefixName` *`prefix`*

Adds a prefix to the inserted cells.

Default: `ECO`

`-refinePlace {true|false}`

Legalizes placement whenever `ecoAddRepeater` or `ecoChangeCell` are used. If `false`, you must run `refinePlace` after ECO if you want to legalize placement.

Note: Specifying `-refinePlace true` can affect performance when large numbers of cells are added or modified.

Default: `true`

`-reset` Resets parameters to their default values. The `-reset` parameter must be the first parameter specified. If you specify reset by itself, the software resets all `setEcoMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-spreadInverter {true|false}`

Distributes an inverter pair evenly between the driver and the first bifurcation point. The design must be routed before you use this parameter.

Default: `true`

`-updateTiming {true|false}`

Controls whether to `eco*` commands recompute timing. If `true`, the eco* commands recompute timing. If `false`, the commands do not recompute timing, and us can later use `report_timing` to recompute timing.

Default: `true`

Examples

- The following command does not legalize placement whenever `ecoAddRepeater` or `ecoChangeCell` are used:

  ```
  setEcoMode -refinePlace false
  ```

- The following command resets the `-honorDontTouch` parameter to its default value:

  ```
  setEcoMode -reset -honorDontTouch
  ```

- The following command resets all `setEcoMode` parameters to their default values:

  ```
  setEcoMode -reset
  ```

14

Low Power Commands

addBufferForFeedThrough

```
addBufferForFeedThrough
     -net net_name
     {-powerDomain domain_name | -bufHInst hier_inst_name}
     -cell cell_name
     -ground (net:pin)
     -power (net:pin)
     -termList {list_of_terminals}
     [-loc x y]
     [-noFixedBuf]
     [-noRefinePlace
     [-prefix prefix]
```

Adds a feedthrough buffer in the power domain for a net crossing a power domain, or in the top module for the net, or in the hierarchical instance specified for the net.

Parameters

`-bufHInst` *`hier_inst_name`*

Specifies the hierarchical instance where the feedthrough buffer is inserted.

`-cell` *`cell_name`* — Specifies the buffer cell to use for the feedthrough.

`-ground (`*`net`*`:`*`pin`*`)` — Specifies the feedthrough buffer's ground connection, which the tool connects to the primary ground.

`-loc` *`x y`* — Specifies the location of the feedthrough buffer.

`-net` *`net_name`* — Specifies the name of the net on which you want to insert a feedthrough buffer.

`-noFixedBuf` — Causes the tool to leave the buffer in an unfixed state. By default, the tool leaves the buffer in a fixed state.

`-noRefinePlace` — Omits placement refinement (refinePlace) after buffer insertion. By default, the tool refines placement.

`-power (`*`net`*`:`*`pin`*`)` — Specifies the feedthrough buffer's power connection, which the tool connects to the primary ground.

`-powerDomain` *`domain_name`*

Specifies the power domain names where the feedthrough is inserted. You cannot specify `-bufHInst` or `-addInTopModule` with this command.

`-prefix` *`prefix`* — Specifies the prefix for the feedthrough buffer name.

`-termlist {`*`list_of_terminals`*`}`

Specifies the list of pins driven by the feedthrough buffer.

Examples

- The following command adds `BUFF1` in power domain `PD1` for `net1`:

  ```
  addBufferForFeedThrough -net net1 -powerDomain PD1 -cell BUFF1
  ```

- The following command adds `BUFF1` to hierarchical instance `Hinst1` for `net1`:

  ```
  addBufferForFeedThrough -net net1 -bufHinst Hinst1 -cell BUFF1
  ```

- The following command adds `BUFF1` in `topmodule` for `net1`:

  ```
  addBufferForFeedThrough -net net1 -addInTopModule -cell BUFF1
  ```

addIsolationCell

```
addIsolationCell
     [-excNet netName]
     [-excNetFile fileName]
     [-prefix prefix]
     [-forceIsoCellInPDHInst | -noForceIsoCellInPDHinst]
     [-noFTermIsoCell]
     [-selNet netName]
     [-selNetFile fileName]
```

Adds isolation cells to nets that cross power domains. You must define the isolation cells in the shifter table before you use this command. By default, the software inserts isolation cells on all nets crossing power domains.

Important

The `addIsolationCell` command is obsolete. To ensure compatibility with future releases, create a Common Power Format (CPF) file and include the related CPF commands.

For backward compatibility, the `addIsolationCell` command still works in this release, but the tool issues a warning message. By default, the design created with this command cannot be saved to the Encounter database. To save an Encounter database, set the following variable:

```
setVar dbgSaveOldMsvDbOnly 1
```

Parameters

`-excNet` *`netName`*
: Adds isolation cells to all nets except the specified net.

`-excNetFile` *`fileName`*
: Specifies the name of the file containing a list of nets for the software to exclude when adding isolation cells. You must list each net on a separate line in the file.

`-forceIsoCellInPDHInst | -noForceIsoCellInPDHinst`
: Forces the software to add isolation cells to each HInst in the location power domain defined in the shifter table. This is the default behavior. To suppress this behavior, use `-noForceIsoCellInPDHinst`.

  Default: `-forceIsoCellInPDHinst`

`-noFTermIsoCell` Prevents the software from adding isolation cells to nets that cross power domains if the net is connected to an fterm.

Default: Adds a isolation cell to a net that traverses power domains and connects to or from an fterm. The software assumes that the fterm is located in the default power domain.

`-prefix` *`prefix`* Adds the specified prefix to the added instance names. This prefix overrides the default prefix.

Default: `isoBuf_`

`-selNet` *`netName`*

Adds isolation cells to the specified nets.

`-selNetFile` *`fileName`*

Specifies the name of the file containing a list of nets for the software to include when adding isolation cells. You must list each net on a separate line in the file.

Command Order

You must first define the isolation cells in the shifter table, then load the shifter table:

`loadShifter -infile` *`shifter_file_name`*

For more information, see "loadShifter" on page 371.

Examples

- The following command adds isolation cells to all nets crossing power domains, as defined in the isolation section of the shifter table.

  ```
  addIsolationCell
  ```

- The following command adds isolation cells to all nets crossing power domains, except for `net1`:

  ```
  addIsolationCell -excNet net1
  ```

- The following command adds isolation cell to all nets crossing power domains, except for the nets included in `excludedNetsFile`:

  ```
  addIsolationCell -excNetFile excludedNetsFile
  ```

- The following command adds isolation cells to nets with the prefix `NET1`:

  ```
  addIsolationCell -prefix NET1
  ```

addPowerSwitch

```
addPowerSwitch
     {-column
     -powerDomain
     -enablePinIn {listOfPins}
     -enableNetIn {listOfNets}
     -enablePinOut {listOfPins}
     -enableNetOut {netBaseName}
     -globalSwitchCellName cell_syntax
     -globalBufferCellName cell_syntax
     -horizontalPitch float
     [-area {x1 y1 x2 y2}]
     [-switchModuleInstance]
     [-bottomOffset float]
     [-leftOffset float]
     [-rightOffset value]
     [-topOffset value]
     [-firstInstanceName cellName]
     [-power {(VDD:VDD1 VDD2...)...}]
     [-ground {(GND:GND1 GND2...)...}]
     [-checkerBoard]
     [-incremental]
     [-width length_in_um]
     [-height length_in_um]
     [-orientation orientation_syntax]
     [-connectBottomSwitchEnablePins {LtoR | RtoL}
     [-parallelEnable]
     [-globalPattern pattern_syntax]
     [-globalBufferCell cell_syntax]
     [-globalBreakerCell cell_syntax]
     [-acknowledgeTreeCell cell_name]
     [-acknowledgeTreeHierInstance hier_instance]
     [-backToBackChain {LtoR | RtoL}]
     [-loopbackAtEnd]
     [-maxChainDepth integer]
     [-snapToNearest]
     [-skipRows]
     [-topDown]
     [-placeByNetWireIntersect {h_net h_layer v_net v_layer}]
     [-placementAdjustX {delta_in_x}]
     [-placementAdjustXY {delta_in_x delta_in_y}]
     [-placementAdjustY {delta_in_y}
     [-reportFile filename]
     |-ring
     -powerDomain powerDomainName
     -enablePinIn {listOfPins}
     -enableNetIn {listOfNets}
     -globalSwitchCellName cell_sytax
     -switchModuleInstance instanceName
     -vertex {x1 y1 x2 y1 x2 y2 x3 y2 x3 y3 x1 y3... xn yn}
```

```
[-enableNetOut {listOfNets}]
[-enablePinOut {listOfPins}]
[-globalBreakerCellName cell_syntax]
[-power {(VDD:VDD1 VDD2...)...}]
[-ground {(GND:GND1 GND2...)...}]
[-startEnableChainAtCorner integer or corner_syntax
[-firstInstanceName instanceName]
[-distribute]
[-noFiller]
[-incremental]
[-counterclockwise]
[-centerTapPins pinname_list]
[-reportFile filename]
[-specifySideList [0|1 ...]
[-bottomSide [0|1]]
[-leftSide [0|1]]
[-horizontalSide [0|1]]
[-rightSide [0|1]]
[-topSide [0|1]]
[-verticalSide [0|1]
[-breakerCellSideList {cell_syntax ...]
[-breakerCellNameBottom cell_syntax]
[-breakerCellNameHorizontal cell_syntax]
[-breakerCellNameLeft cell_syntax]
[-breakerCellNameRight cell_syntax]
[-breakerCellNameTop cell_syntax]
[-breakerCellNameVertical cell_syntax]
[-globalBreakerCellName cell_syntax]
[-bufferCellSideList {cell_syntax ...}]
[-bufferCellNameBottom cell_syntax]}
[-bufferCellNameHorizontal cell_syntax]
[-bufferCellNameLeft cell_syntax]
[-bufferCellNameRight cell_syntax]
[-bufferCellNameTop cell_syntax]
[-bufferCellNameVertical cell_syntax]
[-globalBufferCellName cell_syntax]
[-fillerCellSideList {cell_syntax ...}]
[-fillerCellNameBottom cell_syntax]
[-fillerCellNameHorizontal cell_syntax]
[-fillerCellNameLeft cell_syntax]
[-fillerCellNameRight cell_syntax]
[-fillerCellNameTop cell_syntax]
[-fillerCellNameVertical cell_syntax]
[-globalFillerCellName cell_syntax]
[-switchCellSideList {cell_syntax ...}]
[-switchCellNameBottom cell_syntax]
[-switchCellNameHorizontal cell_syntax]
[-switchCellNameLeft cell_syntax]
[-switchCellNameRight cell_syntax]
[-switchCellNameTop cell_syntax]
[-switchCellNameVertical cell_syntax]
```

```
[-switchCellSideList {cell_syntax ...}
[-cornerCellList {listOfCornerCells}]
[-cornerOrientationList {orientation_syntax ...]
[-insideCornerCellList {cellName ...]
[-r0Corner corner_syntax]
[-r0InsideCorner {LT | TR | RB | BL}
[-r0Side {L | T | R | B}]
[-r0SideOrientation orientation_syntax]
[-switchPitch value]
[-switchPitchBottom value]
[-switchPitchHorizontal value]
[-switchPitchLeft value]
[-switchPitchRight value]
[-switchPitchTop value]
[-switchPitchVertical value]
[-switchPitchSideList {value ...}
[-sideOffsetList {value ...}]
[-bottomOffset float]
[-horizontalOffset float]
[-leftOffset float]
[-rightOffset float]
[-topOffset float]
[-verticalOffset value]
[-sideOffsetList {value ...}]
[-startOffset value]
[-startOffsetBottom value}
[-startOffsetHorizontal value]
[-startOffsetLeft value]
[-startOffsetRight value]
[-startOffsetTop value]
[-startOffsetVertical value]
[-sideStartOffsetList [value...]
[-forceOffset]
[-globalOffset float]
[-endOffset value]
[-endOffsetBottom value]
[-endOffsetHorizontal value]
[-endOffsetLeft value]
[-endOffsetRight value]
[-endOffsetTop value]
[-endOffsetVertical value]
[-sideEndOffsetList {value ...}]
[-sidePatternList {pattern_syntax ...}]
[-bottomPattern pattern_syntax]
[-leftPattern pattern_syntax]
[-horizontalPattern pattern_syntax]
[-rightPattern pattern_syntax]
[-topPattern pattern_syntax]
[-verticalPattern pattern_syntax]
[-globalPattern pattern_syntax]
[-continuePattern]
```

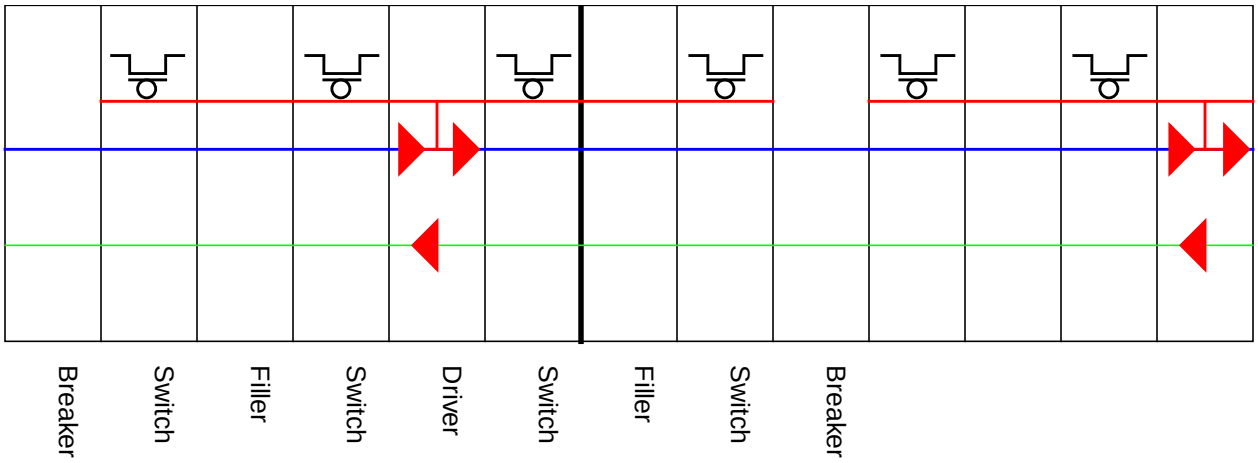
```
[-sideNumSwitchList {integer ...}
[-bottomNumSwitch integer]
[-horizontalNumSwitch value]
[-leftNumSwitch integer
[-rightNumSwitch integer]
[-topNumSwitch integer]
[-verticalNumSwitch value]
[-sideOrientationList {orientation_syntax ...}]
[-bottomOrientation orientation_syntax]
[-horizontalOrientation orientation_syntax]
[-leftOrientation orientation_syntax]
[-rightOrientation orientation_syntax]
[-topOrientation orientation_syntax]
[-verticalOrientation orientation_syntax]}

[{-column | -ring}
[-bufferDelay {value_in_seconds | value_in_seconds value_in_seconds}]
[-cellEM value_in_amps]
[-commitPrototype {0|1)]
[-deviceThresholdVolt value_in_volts]
[-idSat value_in_amps]
[-iLeak value_in_amps]
[-loadCapacitance value_in_farads]
[-maxIRPercent float]
[-maxLeakageCurrent value_in_amps]
[-maxLeakagePercent float]
[-maxRampUpCurrent value_in_amps]
[-maxSwitchIR value_in_volts]
[-noPgDecapEstimate {0|1)]
[-noPgCapacitanceEstimate {0|1)]
[-numberSimultaneousRampUpChain integer]
[-pgCapacitance value_in_farads]
[-pgCapacitanceFactor value_in_farads]
[-pgInductance value_in_henrys]
[-protoReportFile fileName]
[-prototypeChainDepth {0|1)]
[-prototypeChainDepthGivenRampUpCurrent {0|1)]
[-prototypeDelayGivenRampUpCurrent {0|1)]
[-prototypeIgnoreLeakage {0|1)]
[-prototypeMaxChainDepth {0|1)]
[-prototypeMinChainDepth {0|1)]
[-prototypeNumberSwitches {0|1)]
[-prototypeRampUpTime {0|1)]
[-prototypeSweepChainDepth {min_integer max_integer increment_integer}]
[-prototypeSweepSwitchNumber {min_integer max_integer increment_integer}]
[-rampUpChainDepth integer]
[-rampUpCurrent value_in_amps]
[-rampUpRailVoltagePercent float]
[-rampUpTime min_value_in_seconds max_value_in_seconds]
[-readPowerSwitchCell fileName]
[-rOn {value_in_ohms | value_in_ohms value_in_ohms}]
```

```
[-toalLeakagePower value_in_watts]
[-totalPower value_in_watts]
[-voltage value_in_volts] ]
```

Creates columns of power switches inside or a ring of power switches around a specified power domain that shuts off.

The `addPowerSwitch` command supports five type of cells: switch cells, filler cells, breaker cells, corner cells (for the ring style only), and buffer cells. You can define a cell pattern when you create power switch rings.

The following figure shows one possible sequence of cells. The sequence contains cells as follows: breaker, switch, filler, switch, buffer, switch, filler, switch, breaker.



If you create a power switch ring around a power domain, the switches you insert reside in a different power domain. Unless the specified power domain resides inside of a non-default power domain, switches are inserted into the default power domain. The command creates a switch ring on the top, bottom, left, and right sides of the specified power domain unless you assign a specific side.

Before you insert power switches in a ring style, you must assign power switch cells to `CLASS RING` in the `.lef` file and connect power/ground nets and pins of the power switch cells. You must have existing enable nets to drive the buffer inside the power switch cells and acknowledge nets exiting the power switch cell. You need these net names when you specify `-enableNetIn` and `-enableNetOut`.

Before you insert power switches in a column style, you must assign power switch cells to `CLASS CORE` and the correct `SITE` in the `.lef` file and connect power/ground nets and pins of the power switch cells. The distance between columns of switches (the horizontal pitch

value) must be in µm. You might need to use `createShifterRows` to create a new set of shifter rows based on the SITE name if the power switch cell in unique.

The column style is favorable for channelless designs. In this case, all blocks are abutted and there is no space for rings. In ring style, you must have a space greater than the max ring cell height between the power domain and the adjacent power domain. This space is primarily occupied by ring switch cells and power routing.

If you create power switch columns within a specified power domain the distance between the columns of switches is determined by the `-horizontalPitch`. The first (left-most) column of switches is located `-leftOffset` from the power domain's left boundary. The `-enableNetIn`, which might consist of one or more enable nets, drives the `-enablePinIn` of the first switch cell of each column.

In column style, nets are assigned to pins sequentially. For example, if the list consists of three nets, and there are five columns of switches, net `1` drives the pin of the first switch in column `1` and `4`, net `2` connects to column `2` and `5`, and net `3` only drive the `-enablePinIn` of the first switch in column `3`.

Rectilinear power domains are supported in both ring-style and column-style switch insertion.

Regardless of whether you insert columns or a ring, the software ties the `-enablePinOut` of the first switch cell to the `-enablePinIn` of the next switch cell in the same column or on the same side of a ring. All the enable pin of the switch cells in the same column/side are daisy-chained together. For the last switch of the column, the `-enablePinOut` is tied to `-enableNetOut`.

Syntax Conventions:

- Cell syntax

  `cell_syntax`

  `cell` (single cell)

  {*`cell1 cell2 …`*} (list of cells)

  {{*`cell id`*}} (cell with id assigned)

  {{cell1 id1} {cell2 id2} …} (cell list with id assigned to each cell)

- Orientation syntax

  `orientation_syntax`

  R0|R90|R180|R270|MX|MY|MX90|MY90

- Corner syntax (Ring style only)

  corner_syntax

  LT|TR|RB|BL

- Pattern syntax (Ring style only)

  pattern_syntax

  {id1 id2 {id * n}} (Example: n=integer -> repeat id n times)

  - id, id1, id2 are defined in *cell_syntax*
  - Default id is the same as the cell name if *cell_syntax* does not define it.

For example:

Power switch cell name: CDN_SWT_CELL

Filler cell name: CDN_FLL_CELL

Buffer cell name: CDN_BUF_CELL

Breaker cell name: CDN_BRE_CELL

cell_syntax

cell (single cell): CDN_SWT_CELL

{*cell1 cell2*}: {CDN_SWT_CELL1X CDN_SWT_CELL2X …}

{{*cell id*}}: {CDN_SWT_CELL S} <- S can be any alphabetic character from A to Z or a to z, or any combination of alphabetic characters or numerals. The id must always start with an alphabetic character.

Precedence Rules

For the -*direction*Side parameters, the software applies their settings in the following precedence rules:

- -specifySides
- -topSide, -rightSide, -leftSide, -bottomSide
- -horizontalSide, -verticalSide

Higher precedence parameters override lower precedent parameters.

Parameters

`-acknowledgeTreeCell` *`cell_name`*

Specifies the name of cell to use in acknowledge tree synthesis. If you specify this parameter but do not specify `-acknowledgeTreeHierInstance`, the tool builds the acknowledge tree at the top level of the design. This parameter provides control over the time it takes to switch a power domain off and on.

Note: This parameter applies to power switch column creation only.

`-acknowledgeTreeHierInstance` *`hier_instance`*

Inserts an enable net out signal tree of specified cell instances in the specified hierarchical instance in the design. This parameter provides control over the time it takes to switch a power domain off and on.

Note: This parameter applies to power switch column creation only.

`-area {`*`x1 y1 x2 y2`*`}`

Defines an area within a power domain where the tool can insert power switches.

`x1 y1` is the lower left coordinate of the area.

`x2 y2` is the upper right corner of the area.

You can specify `-leftOffset`, `-rightOffset`, `-topOffset`, or `-bottomOffset` to offset the switches from the boundaries of the area.

Note: This parameter applies to power switch column creation only.

`-backToBackChain {LtoR | RtoL}`

Connects the `enableNetOut` at the top of a column to the `enableNetIn` at the top of the next column, and connects the `enableNetOut` at the bottom of the column to the `enableNetIn` at the bottom of the next column. The chain proceeds left to right or right to left. By default, the `enableNetIn` is connected to the bottom of each column, and the `enableNetOut` exits from the top of each column, in parallel.

This parameter is optional.

Note: This parameter applies to power switch column creation only.

`-bottomNumSwitch` *`integer`*

Specifies the number of switches to insert on the bottom side(s) of the power domain.

Note: This parameter applies to power switch ring creation only.

Three scenarios can occur:

- If the spaces available match the number specified, the software fills these spaces with switches.
- If the number exceeds the spaces available, the software issues a warning message.
- If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space.

Also, if you specify `f` instead of an integer, the tool automatically fills the bottom side with switches and filler cells.

This parameter is optional.

See `-horizontalNumSwitch`, `-verticalNumSwitch`, `-leftNumSwitch`, `-rightNumSwitch`, and `-topNumSwitch`.

Default: `f`

`-bottomOffset` *`float`*

Specifies the minimum distance between the bottom power domain boundary and the nearest switch cell.

This parameter can be used in either of two ways:

- Columns

  Specifies the distance between the bottom-most switch in the switch column and the bottom fence boundary of the power domain. You can use this parameter along with `-width`, `-height`, and other offsets to define a unique, specific area within the power domain for placing switches. By default, the switch column can abut the bottom fence boundary.

- Rings

  Specifies distance from the bottom edge outside of the power domain fence and the bottom side(s) of the power switch ring. See `-horizontalOffset, -verticalOffset, -leftOffset`, `-rightOffset`, `and -topOffset`.

  Note: When you create a power domain, a `minGap` exists for this power domain, and the switches are inserted within this `minGap`. However, if the specified `minGap` is less than the switch dimension, then the tool automatically extends the `minGap` for this power domain.

For more information, see “Using Pitch Control and Offsets” in the “Low Power Design” chapter of the *Encounter User Guide*.

Default: `0`

`-bottomOrientation` *`orientation_syntax`*

Specifies the cell orientation for all bottom side(s) of the power switch ring.

`-bottomPattern` *`pattern_syntax`*

Defines a pattern for the bottom side(s) of the ring only.

Note: This parameter applies to power switch ring only.

`-bottomSide [0 | 1]`

Inserts switches on the bottom side(s) of the power domain. If you do not specify a side, the software creates a full power switch ring around the specified power domain. If you specify a side, the software inserts switches on that side. In this case, the tool sets `-bottomSide` to `1` and sets the other sides to `0`.

You can specify multiple sides and the software inserts switches only on the sides you specify. The tool sets the specified sides to `1` and sets the unspecified sides to `0`.

Note: This parameter applies to power switch ring creation only.

Default: `1`

`-breakerCellNameBottom` *`cell_syntax`*

Specifies the breaker cells for the software to use when terminating buffer signal propagation on the bottom side(s) of the ring.

Note: This parameter applies to power switch ring creation only.

`-breakerCellNameHorizontal` *`cell_syntax`*

Specifies the breaker cells for the software to use when terminating buffer signal propagation on the horizontal side(s) of the ring.

Note: This parameter applies to power switch ring creation only.

`-breakerCellNameLeft` *`cell_syntax`*

Specifies the breaker cells for the software to use when terminating buffer signal propagation on the left side(s) of the ring.

Note: This parameter applies to power switch ring creation only.

`-breakerCellNameRight` *`cell_syntax`*

Specifies the breaker cells for the software to use when terminating buffer signal propagation on the right side(s) of the ring.

Note: This parameter applies to power switch ring creation only.

`-breakerCellNameTop` *`cell_syntax`*

Specifies the breaker cells for the software to use when terminating buffer signal propagation on the top side(s) of the ring.

Note: This parameter applies to power switch ring creation only.

`-breakerCellNameVertical` *`cell_syntax`*

Specifies the breaker cells for the software to use when terminating buffer signal propagation on the vertical side(s) of the ring.

Note: This parameter applies to power switch ring creation only.

`-breakerCellSideList {`*`cell_syntax`* `…}`

Specifies the breaker cells to insert on each side of a power domain. The first member of the list indicates the first side relative to Corner 0, which is the lower left corner of the ring. If `-counterclockwise` is specified, the first side is counterclockwise from Corner 0.

The following example shows the breaker cells specified for an eight-sided power switch ring:

```
-specifySideList {1 0 1 0 1 0 1 0}
-breakerCellSideList {cell1 cell1 cell2 cell2 cell3 \ cell3
cell4 cell4}
```

`cell1` is used for the first side. You can then specify `cell1` again to provide a valid cell type, even though there can be no cells on the second side.

Important

You must include a space between each item in the list.

You *must* specify a valid cell name for each side, even if no cells can be specified for a side.

The tools checks that the number of cells you specify is the same number you specified with `-specifySideList`, and issues an error message if it finds a discrepancy.

Note: This parameter applies to power switch ring creation only.

`-bufferCellNameBottom` *`cell_syntax`*

Specifies the buffer cells to use on the bottom side(s) of the switch ring.

Note: This parameter applies to power switch ring creation only.

`-bufferCellNameHorizontal` *`cell_syntax`*

Specifies the buffer cells to use on the horizontal side(s) of the switch ring.

Note: This parameter applies to power switch ring creation only.

`-bufferCellNameLeft` *`cell_syntax`*

Specifies the buffer cells to use on the left side(s) of the switch ring.

Note: This parameter applies to power switch ring creation only.

`-bufferCellNameRight` *`cell_syntax`*

Specifies the buffer cells to use on the right side(s) of the switch ring.

Note: This parameter applies to power switch ring creation only.

`-bufferCellNameTop` *`cell_syntax`*

Specifies the buffer cells to use on the top side(s) of the switch ring.

Note: This parameter applies to power switch ring creation only.

`-bufferCellNameVertical` *`cell_syntax`*

Specifies the buffer cells to use on the vertical side(s) of the switch ring.

This parameter applies to power switch ring creation only.

`-bufferCellSideList {`*`cell_syntax …`*`}`

Specifies the buffer cells to insert on each side of a power domain. The first member of the list indicates the first side relative to Corner 0, which is the lower left corner of the ring. If `-counterclockwise` is specified, the first side is counterclockwise from Corner 0.

The following example shows the buffer cells specified for an eight-sided power switch ring:

```
-specifySideList {1 0 1 0 1 0 1 0}
-bufferCellSideList {cell1 cell1 cell2 cell2 cell3 cell3 cell4
cell4}
```

`cell1` is used for the first side. You can then specify `cell1` again to provide a valid cell type, even though there can be no cells on the second side.

You must include a space between each item in the list.

You *must* specify a valid cell name for each side, even if no cells can be specified for a side.

The tools checks that the number of cells you specify is the same number you specified with `-specifySideList`, and issues an error message if it finds a discrepancy.

Note: This parameter applies to power switch ring creation only.

`-bufferDelay {`*`value_in_seconds`* `|` *`value_in_seconds value_in_seconds`*`}`

Specifies the delay of the enable buffer in switch. If the switch is a mother-daughter type (with two enables), you can specify two values corresponding to the stage-1 and stage-2 enable buffers respectively.

`-cellEM` *`value_in_amps`*

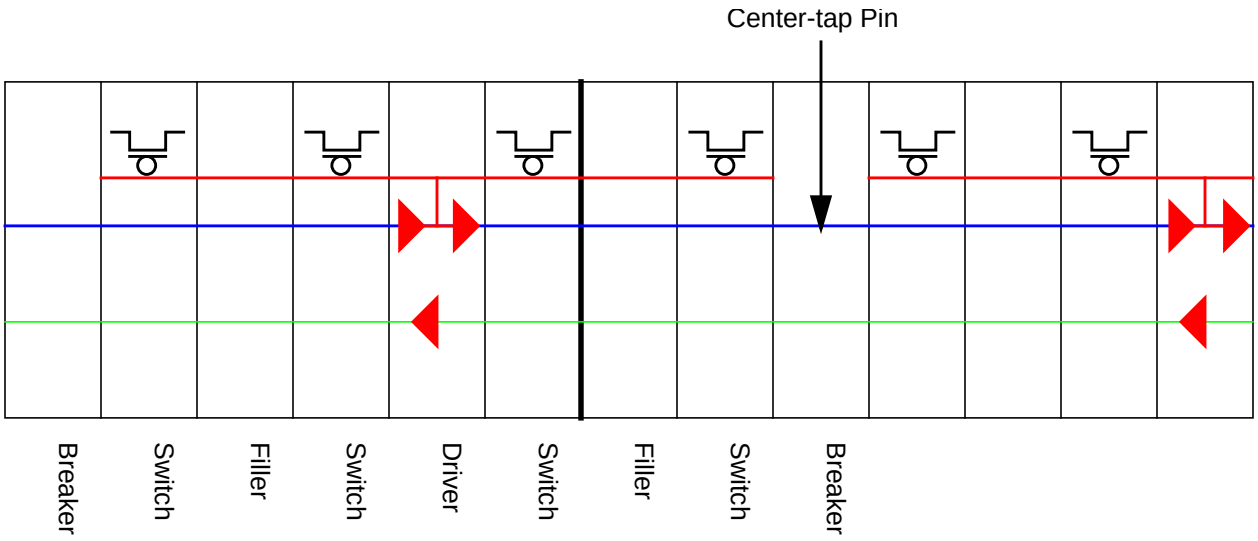
Specifies the maximum current that can pass through the switch due to the electromigration limit of the cell.

`-centerTapPins` *`pinname_list`*

Enables the center tap capability, and specifies the pins to use as center taps. You must use breaker cells to terminate the buffer signal propagation on both sides of the center tap buffer.

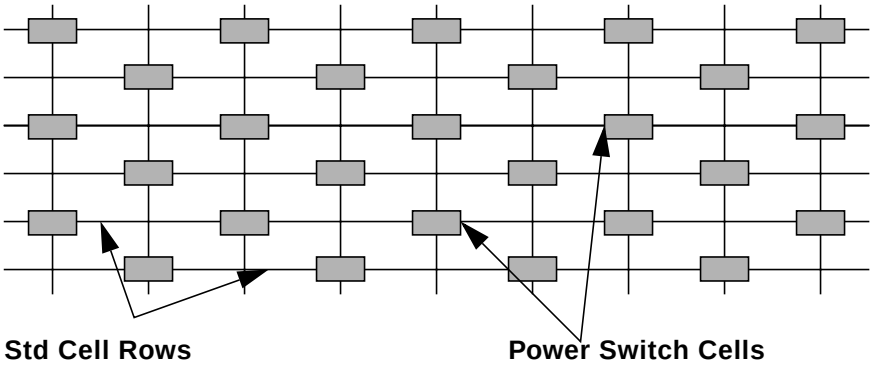
The following figure shows that, depending on the specified pin name list, the software assigns a pin name to the enable net. This feature can be useful for debugging the enable signal.

Note: This parameter applies to power switch ring creation only.



`-checkerBoard` Arranges switch cells in a checkerboard pattern within a power domain.

The following figure shows the checkerboard pattern for power switch cells in the column style of switch cell insertion.



Note: This parameter applies to power switch column creation only.

`-commitPrototype {1|0}`

Adds the number of switches determined by `-prototypeNumberSwitches`. This parameter does not work with any other prototyping commands.

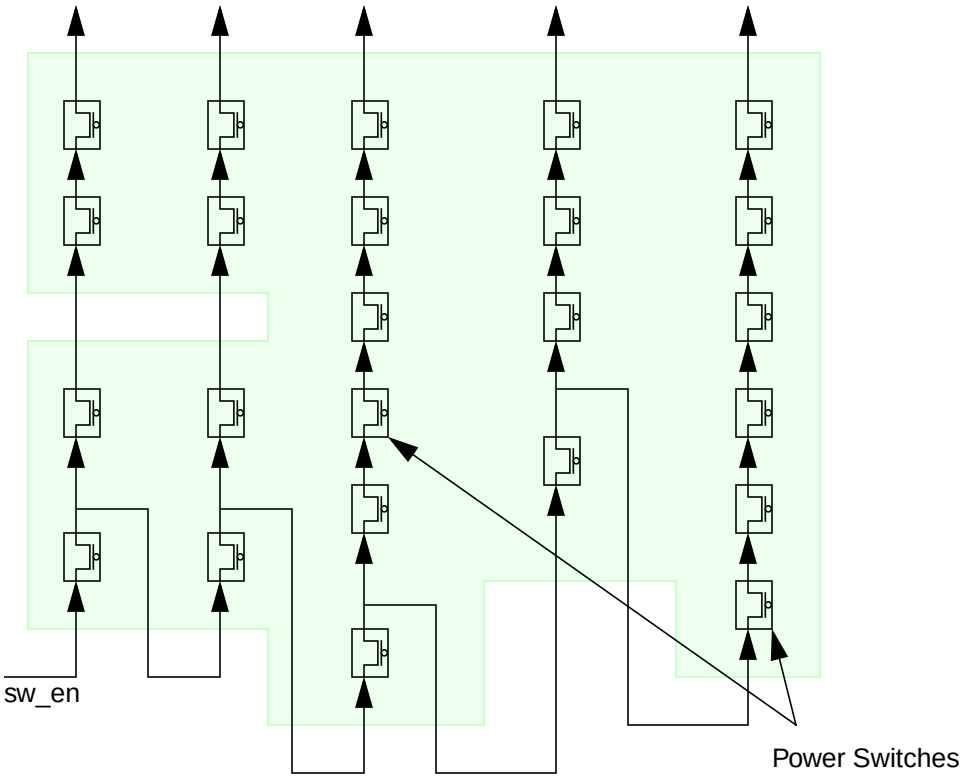
Default: If you specify `-commitPrototype` but do not specify a value, the default is `1`. If you do not specify -commitPrototype, the default is `0`.

Note: This parameter applies to power switch column creation only.

`-connectBottomSwitchEnablePins {LtoR | RtoL}`

Connects the first switch of the first column to the enable net specified. The software connects the first switch in the remaining columns to the output enable net of the first switch in the previous column.

The following figure illustrates the `LtoR` option (default) for a rectilinear power domain.



Note: This parameter applies to power switch column creation only.

Default: The enable pin of the first switch of each column is connected directly to the enable net signal specified.

`-continuePattern`

Continues a specified pattern from one side of the ring to the next side. You must also specify `-pattern` when you use this parameter.

Note: This parameter applies to power switch ring creation only.

Default: If you do not use this parameter, the software stops the pattern at the end of a side and restarts the pattern from the beginning on the next side.

`-counterclockwise`

Places the cell pattern in a counterclockwise direction in a switch ring. Switch placement begins at the lower left corner of the ring.

Note: This parameter applies to power switch ring creation only.

Default: If you do not specify this parameter, the software places the pattern in a clockwise direction.

`-cornerCellList {`*`listOfCornerCells`*`}`

Specifies the list of available corner cells that can be used by the tool to fill the corners of the ring. This parameter is optional.

If user specifies only one corner cell, that same cell is

used on all four corners with the appropriate orientation.

Note: This parameter applies to power switch ring creation only.

`-cornerOrientationList {`*`orientation_syntax`* `…}`

Specifies cell orientations for the tool to use for each corner of switch ring, starting with Corner 0 (the lower left corner). List the orientations in a list, with the first orientation corresponding to the first side, the second orientation to the second side, and so on. If you do not specify this parameter, the software determines the corner cell orientations.

`-deviceThresholdVolt` *`value_in_volts`*

Specifies the device threshold voltage for ramp up prototyping. If you do not specify this parameter, the tool computes a default value based on the switch characterization parameters.

`-distribute` Distributes switches of different types as evenly as possible on a ring side. For example, if there are 40 available slots, and there are 20 switches and 20 fillers, the software will alternate switches and fillers in the available slots. Use this option together with `-horizontalNumSwitch`, `-verticalNumSwitch`, `-leftNumSwitch`, `-rightNumSwitch`, `-topNumSwitch`, or `-bottomNumSwitch`.

`-enableNetIn {listOfnets}`

Specifies the list of input nets that the software attaches to the enable input pins. The order in which you specify these nets is important because nets are assigned to pins sequentially.

If you have already specified `create_power_domain -shutoff_condition` in your CPF file, you do not need to specify this parameter. Enable nets and pins are connected automatically if they reside at the same level in the hierarchy.

If you specify `-enableNetIn`, it will override `create_power_domain -shutoff_condition`.

Note: This parameter is required for both power switch ring and column creation.

`-enableNetOut {`*`netBaseName`* `| {listOfNets}}`

Specifies the enable output nets that the software attaches to the enable output pins. The order in which you specify these nets is important because nets are assigned to pins sequentially. For the last switch of the ring, the `-enablePinOut` is tied to `-enableNetOut`.

Note: You must specify *`netBaseName`* for columns or a list of nets for rings.

Note: This parameter is required power switch column creation. It is optional for power ring creation.

`-enablePinIn {listOfPins}`

Specifies the list of enable input pins of the buffers inside the switch cell. The order in which you specify these pins is important because nets are assigned to pins sequentially. The software ties the `-enablePinOut` of the first switch cell to the `-enablePinIn` of the next switch cell in the same ring: All the enable pin of the switch cells in the same column are daisy-chained together. For the first switch of the ring, the `-enablePinIn` is tied to `-enableNetIn`.

Note: This parameter is required for both power switch ring and column creation.

`-enablePinOut {listOfPins}`

Specifies the list of enable output pin(s) of the buffer(s) inside the switch cell. The order in which you specify these pins is important because nets are assigned to pins sequentially. The software ties the `-enablePinOut` of the first switch cell to the `-enablePinIn` of the next switch cell in the same column: All the enable pin of the switch cells in the same column are daisy-chained together. For the last switch of the column, the `-enablePinOut` is tied to `-enableNetOut`.

Note: This parameter is required for both power switch ring and column creation.

`-endOffset` *`value`*

Places the last switch on every side of the power domain at the specified distance from the end of each side.

The side end is not necessarily determined by the power domain boundary. You can use `-leftOffset`, `-rightOffset`, `-topOffset`, `-bottomOffset`, `-horizontalOffset`, and/or `-verticalOffset` to determine the side end relative to the power domain boundary.

For more information, see “Using Pitch Control and Offsets” in the “Low Power Design” chapter of the *Encounter User Guide*.

Default: 0

`-endOffsetBottom` *`value`*

Places the last switch on bottom side of the power domain at the specified distance from end of the side.

The side end is not necessarily determined by the power domain boundary. You can use `-leftOffset`, `-rightOffset`, and/or `-verticalOffset` to determine the side end relative to the power domain boundary.

Ring style:

- If the value is positive, the software places the switch to the right of the end location, or to the left of the end location if `-counterclockwise` is specified.
- If the value is negative, the software places the switch to the left of the end location, or to the right of the end location if `-counterclockwise` is specified.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

`-endOffsetHoriztonal` *`value`*

Places the last switch on the horizontal side of the power domain at the specified distance from end of the side.

The side end is not necessarily determined by the power domain boundary. You can use `-leftOffset`, `-rightOffset`, and/or `-verticalOffset` to determine the side end relative to the power domain boundary.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

`-endOffsetLeft` *`value`*

Places the last switch on the left side of the power domain at the specified distance from end of the side.

The side end is not necessarily determined by the power domain boundary. You can use `-topOffset` `-bottomOffset` and/or `-horizontalOffset` to determine the side end location relative to the power domain boundary.

Ring style:

- If the value is positive, the software places the switch below the end location, or above the end location if `-counterclockwise` is specified.
- If the value is negative, the software places the switch above the end location, or above the end location if `-counterclockwise` is specified.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

`-endOffsetRight` *`value`*

Places the last switch on the right side of the power domain at the specified distance from end of the side.

The side end is not necessarily determined by the power domain boundary. You can use the `-topOffset`, `-bottomOffset`, or `-horizontalOffset` to determine the side end relative to the power domain boundary.

Ring style:

- If the value is positive, the software places the switch above the end location, or below the end location if `-counterclockwise` is specified.
- If the value is negative, the software places the switch below the end location, or below the end location if `-counterclockwise` is specified.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

`-endOffsetTop` *`value`*

Places the last switch on the top side of the power switch ring at the specified distance from end of the side.

The side end is not necessarily determined by the power domain boundary. You can use `-leftOffset`, `-rightOffset` or `-verticalOffset` to determine the side end relative to the power domain boundary.

Ring style:

- If the value is positive, the software places the switch to the left of the end location, or to the right of the end location if `-counterclockwise` is specified.
- If the value is negative, the software places the switch to the right of the end location, or to the left of the end location if `-counterclockwise` is specified.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

`-endOffsetVertical` *`value`*

Places the last switch on the vertical sides of the power switch ring at the specified distance from end of the side.

The side end is not necessarily determined by the power domain boundary. You can use `-topOffset`, `-bottomOffset`, and/or `-horizontalOffset` to determine to the side end relative to the power domain boundary.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

`-fillerCellNameBottom` *`cell_syntax`*

Specifies the filler cell used for the bottom side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional.

Note: This parameter applies to power switch ring creation only.

Default: The filler cell specified by `-globalFillerCellName`.

`-fillerCellNameHorizontal` *`cell_syntax`*

Specifies the filler cell used for the horizontal side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional.

Note: This parameter applies to power switch ring creation only.

Default: The filler cell specified by `-globalFillerCellName`.

`-fillerCellNameLeft` *`cell_syntax`*

Specifies the filler cell used for the left side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional.

Note: This parameter applies to power switch ring creation only.

Default: The filler cell specified by `-globalFillerCellName`.

`-fillerCellNameRight` *`cell_syntax`*

Specifies the filler cell used for the right side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional.

Note: This parameter applies to power switch ring creation only.

Default: The filler cell specified by `-globalFillerCellName`.

`-fillerCellNameTop` *`cell_syntax`*

Specifies the filler cell used for the top side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional.

Note: This parameter applies to power switch ring creation only.

Default: The filler cell specified by `-globalFillerCellName`.

`-fillerCellNameVertical` *`cell_syntax`*

Specifies the filler cell used for the vertical side(s) of the ring. You can specify a different filler cell for each side of the ring. This parameter is optional.

Note: This parameter applies to power switch ring creation only.

Default: The filler cell specified by `-globalFillerCellName`.

`-fillerCellSideList {`*`cell_syntax`* `…}`

Specifies the filler cells to insert on each side of a power domain. The first member of the list indicates the first side relative to Corner 0, which is the lower left corner of the ring. If `-counterclockwise` is specified, the first side is counterclockwise from Corner 0.

The following example shows the filler cells specified for an eight-sided power switch ring:

```
-specifySideList {1 0 1 0 1 0 1 0}
-fillerCellSideList {cell1 cell1 cell2 cell2 cell3 \ cell3 cell4
cell4}
```

`cell1` is used for the first side. You can then specify `cell1` again to provide a valid cell type, even though there can be no cells on the second side.

Important

You must include a space between each item in the list.

You *must* specify a valid cell name for each side, even if no cells can be specified for a side.

The tools checks that the number of cells you specify is the same number you specified with `-specifySideList`, and issues an error message if it finds a discrepancy.

Note: This parameter applies to power switch ring creation only.

`-firstInstanceName` *`instanceName`*

Specifies the instance name of an existing switch cell in the netlist to use as the first cell in the column or ring. Usually, there is at least one switch instantiated in a module (specified by `-switchModuleInstance`) in the netlist. The software creates a unique instance name for each switch inserted. This parameter is optional. If the netlist does not contain switches, then this parameter is invalid.

Note: This parameter applies to both power switch ring and column creation.

`-forceOffset` Forces the software to place ring switches at the exact offset you specify for all sides. Sides with unspecified offsets are forced to 0.0 µ.

When you specify any other offset without specifying `-forceOffset`, the tool places switches at the smallest distance that is greater than or equal to the specified offset for each side, in order to maintain a contiguous ring.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: Using this parameter might result in gaps in the ring.

Note: This parameter applies to power switch ring creation only.

`-globalBreakerCellName` *`cell_syntax`*

Specifies the cell name of the breaker cell to insert in the power switch ring surrounding the specified power domain, or in the columns inside the power domain.

Note: This parameter applies to both power switch ring and column creation.

`-globalBufferCellName` *`cell_syntax`*

Specifies the cell name of the buffer cell to insert in the power switch ring surrounding the specified power domain, or in the columns inside the power domain, if there are spaces available.

Note: This parameter applies to both power switch ring and column creation.

`-globalFillerCellName` *`cell_syntax`*

Specifies the cell name of the filler cell to insert in the power switch ring surrounding the specified power domain, or in the columns inside the power domain, if there are spaces available.

Note: This parameter applies to both power switch ring and column creation.

`-globalOffset` *`value`*

Offsets all sides of the power switch ring the specified distance from the power domain boundary. Specify the offset in μm.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

Note: This parameter applies to power switch ring creation only.

`-globalPattern` *`pattern_syntax`*

Specifies a pattern of cells to repeat either in the switch rings or switch columns. The pattern consists of a group of different types of cells in a specified order. In the ring style, you can use the `-continuePattern` parameter to continue the pattern around the corner, from one side of the ring to another.

Note: This parameter applies to both power switch ring and column creation.

`-globalSwitchCellName` *`cellName`*

Specifies the cell name of the switch cell to insert in the power switch ring surrounding the specified power domain, or in the column inside the power domain. This parameter is required.

Note: This parameter applies to both power switch ring and column creation.

`-ground` {(*`GND:GND1 GND2`*...)...}

Specifies a list of ground nets and pin connections. If `-ground` has multiple ground nets, the software uses the last one in the list for tie-hi/lo connections. For example, specify `-ground (VSS:gnd vss!)` to indicate that the `gnd` and `vss!` pins of all the instances in a given power domain are connected to the VSS net.

Note: This parameter applies to both power switch ring and column creation.

`-height` *`length_in_um`*

Specifies the height of an area inside the power domain where switches can be placed. You can use this parameter along with `-width`, `-bottomOffset`, and `-leftOffset` to define a unique, specific area in the power domain for placing switches.

Note: This parameter applies to power switch column creation only.

`-horizontalNumSwitch` *`value`*

Specifies the number of switches to insert on the horizontal side(s) of the power domain.

Note: This parameter applies to power switch ring creation only.

Three scenarios can occur:

- If the spaces available match the number specified, the software fills these spaces with switches.
- If the number exceeds the spaces available, the software issues the following error message.
- If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space.

This parameter is optional. See `-leftNumSwitch`, `-rightNumSwitch`, and `-topNumSwitch`.

`-horizontalPattern` *`pattern_syntax`*

Specifies a pattern of cells to repeat on the horizontal side(s) of the power domain when switches are inserted. The pattern consists of a group of different types of cells in a specified order.

Note: This parameter applies to power switch ring creation only.

`-horizontalOffset` *`value`*

Offsets the switch ring from the horizontal edge of the power domain. Specify the offset in μm.

For more information, see “Using Pitch Control and Offsets” in the “Low Power Design” chapter of the *Encounter User Guide*.

Default: `0`

`-horizontalOrientation` *`orientation_syntax`*

Specifies the cell orientation for all horizontal side(s) of the power switch ring.

Note: This parameter applies to power ring creation only.

`-horizontalPitch` *`float`*

Specifies the distance between switch columns in μm. This parameter is required.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch column creation only.

`-horizontalSide [0|1]`

Inserts switches on all horizontal side(s) of a power domain. You cannot specify this parameter in conjunction with `-verticalSide`, `-bottomSide`, `-topSide`, `-leftSide`, or `-rightSide`.

Note: This parameter applies to power switch ring creation only.

Default: `1`

`-idsat {`*`value_in_amps | value_in_amps value_in_amps`*`}`

Specifies the switch current when the switch is operated in saturation mode. If the switch is a mother-daughter type (with two enables), then you can specify two values corresponding to the stage-1 and stage-2 enables respectively.

`-ileak` *`value_in_amps`*

Specifies the switch cell leakage current when the switch is off.

`-incremental` Inserts switches to the power domain without removing the existing switches.

Note: This parameter applies to both power switch ring and column creation.

Default: Running the `addPowerSwitch` command again removes the existing placed switches and inserts new switches.

`-insideCornerCellsList {`*`cellName`* `…}`

Specifies a list of cells for the tool to choose for the inside corners of the power domain.

The following example specifies the corner cells any power switch ring:

```
-insideCornerCellsList {cell1 cell2}
```

Note: You do not need to list a specific cell for a specific corner. The tool selects from the list you provide.

Note: This parameter applies to power switch ring creation only.

`-leftNumSwitch` *`integer`*

Specifies the number of switches to insert on left side(s) of the power domain.

Note: This parameter applies to power switch ring creation only.

Three scenarios can occur:

- If the spaces available match the number specified, then the software fills these spaces with switches.
- If the number exceeds the spaces available, the software issues a warning message.
- If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space.

This parameter is optional. See `-bottomNumSwitch`, `-rightNumSwitch`, and `-topNumSwitch`.

Default: `f`

`-leftOffset` *`float`*

Specifies the minimum distance between the left power domain boundary and the nearest switch cell. See `-bottomOffset`, `-rightOffset`, `-topOffset`, `-verticalOffset`, and `-horizontalOffset`.

- Columns

  Specifies the distance between the left-most switch in the switch column and the right fence boundary of the power domain. You can use this parameter along with `-width`, `-height`, and `-bottomOffset` to define a unique, specific area within the power domain for placing switches. By default, the switch column can abut the right fence boundary.

- Rings

  Specifies distance from the right edge outside of the power domain fence and the right side of the power switch ring. See `-horizontalOffset`, `-verticalOffset`, `-rightOffset`, `-bottomOffset`, and `-topOffset`.

  Note: When you create a power domain, a `minGap` exists for this power domain, and the switches are inserted within this `minGap`. However, if the specified `minGap` is less than the switch dimension, then the tool automatically extends the `minGap` for this power domain.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`
The switches abut the power domain fence if possible.

`-leftOrientation` *`orienation_syntax`*

Specifies the cell orientation for all left side(s) of the power switch ring.

Note: This parameter applies to power ring creation only.

`-leftPattern` *`pattern_syntax`*

Defines a pattern for the left side(s) of the ring only. You can use this parameter with `-bottomPattern`, `-topPattern`, and `-rightPattern`.

Note: This parameter applies to power switch ring creation only.

`-leftSide [0 | 1]`

Inserts switches on the left side(s) of the power domain. If you do not specify a side, the software creates a full power switch ring around the specified power domain. If you specify a side, the software inserts switches on that side. In this case, the tool sets `-leftSide` to `1` and sets the other sides to `0`.

You can specify multiple sides and the software inserts switches only on the sides you specify. The tool sets the specified sides to `1` and sets the unspecified sides to `0`.

Note: This parameter applies to power switch ring creation only.

Default: `1`

`-loadCapacitance` *`value_in_farads`*

Specifies the total capacitive load of the domain for ramp-up analysis.

`-loopbackAtEnd`

Connects the `enablePinOut` of the last cell in the chain to the `enablePinIn` of the same cell.

Note: This parameter applies to column switch creation only.

`-maxChainDepth` *`integer`*

Specifies the maximum number of instances in the enable chain. After the specified maximum, addPowerSwitch starts a new enable chain from the next power switch cell. This feature provides fine control over the time it takes to switch a power domain off and on.

Note: This parameter applies to power switch column creation only.

`-maxIRPercent` *`float`*

Specifies the maximum IR drop allowed inside power domain.

Default: 2 percent

`-maxLeakageCurrent` *`value_in_amps`*

Specifies the maximum leakage current constraint of the power domain.

`-maxLeakagePercent` *`float`*

Specifies the maximum domain leakage constraint as a percentage of the total domain leakage current.

Default: `50` percent

`-maxRampUpCurrent` *`value_in_amps`*

Specifies the maximum ramp up current value used in chain depth and switch delay prototyping. Use this parameter with `-prototypeChainDepthGivenRampUpCurrent` or `-prototypeDelayGivenRampUpCurrent`.

`-maxSwitchIr` *`value_in_volts`*

Specifies the maximum voltage drop allowed across the switch.

`-noFiller` Suppresses the addition of filler cells in power switch rings.

Note: This parameter applies to power switch ring creation only.

`-noPgCapacitanceEstimate {0|1}`

Omits power grid capacitance estimation during ramp up analysis.

Default: `0`

`-noPgDecapEstimate {0|1}`

Omits decoupling capacitance estimation during ramp up analysis.

Default: `0`

`-numberSimultaneousRampUpChain` *`integer`*

Specifies the number of simultaneous ramp up chains for ramp up analysis.

`-orientation` *`orientation_syntax`*

Specifies the orientation of the switch to be placed in the row. The software checks the specified orientation of the switch against the legal orientation of the row in which the switch is to be placed. The software uses the row site information and the row orientation to determine whether the specified orientation is legal.

Default: Follows the row orientation.

Note: This parameter applies to power switch column creation only.

`-parallelEnable`

Connects a power switch's input pin in parallel style rather than daisy-chaining. You do not have to specify `-enablePinOut` or `-enableNetOut`.

Note: This parameter applies to both power switch ring and column creation.

`-perSideNumSwitches` *`value`*

Specifies the number of switches to use on all sides of a power switch ring.

Note: This parameter applies to power switch ring creation only.

`-pgCapacitance` *`value_in_farads`*

Specifies the total power grid capacitance for ramp up analysis.

`-pgCapacitanceFactor` *`float`*

Specifies the ratio of the power grid capacitance as a factor of the total load capacitance.

`-pgInductance` *`value_in_henrys`*

Specifies the effective power grid inductance associated with the power grid and package.

`-placeByNetWireIntersect {`*`h_net h_layer v_net v_layer`*`}`

Places switches at the intersection of specified horizontal net and vertical net on the respective LEF layers of a pre-existing power grid. You must specify the horizontal net and layer first, then the vertical net and layer.

By default, the switches are placed at the bottom-left intersection of the grid. You can offset the switches from the grid intersection by specifying `-placementAdjustX`, `-placementAdjustXY`, or `-placementAdjustY` in addition to this parameter.

Note: This parameter applies to power switch column creation only.

`-placementAdjustX {`*`delta_in_x`*`}`

Specifies the horizontal offset (in micrometers) from the grid intersection specified by `-placeByNetWireIntersect`. The value can be positive or negative.

Note: This parameter applies to power switch column creation only.

`-placementAdjustXY {`*`delta_in_x delta_in_y`*`}`

Specifies the x-y offset (in micrometers) from the grid intersection specified by `-placeByNetWireIntersect`. The values can be positive or negative.

Note: This parameter applies to power switch column creation only.

`-placementAdjustY {`*`delta_in_y`*`}`

Specifies the vertical offset (in micrometers) from the grid intersection specified by `-placeByNetWireIntersect`. The values can be positive or negative.

Note: This parameter applies to power switch column creation only.

`-power` {(*`VDD:VDD1 VDD2`*...)...}

Specifies a list of power nets and pin connections.

For example, specify `-power (Vdd1:Vdd Pwr) (Vdd2:VDD vdd!)` to indicate that the `Vdd` and `Pwr` pins of all the instances in a given power domain are connected to the `Vdd1` net, and the `VDD` and `vdd!` pins are connected to the `Vdd2` net.

If `-power` has multiple power nets, the software uses the last one in the list for tie-hi/lo connections. For example, if `-power {(vdd1:VDD1)(vdd2:VDD2)}` is specified, then the software uses net `vdd2` for tie-hi connection.

Note: You must specify both `-ground` and `-power` to ensure that the instance being moved contains the correct `(net:pin)` connections in the new power domain.

Note: This parameter applies to both power switch ring and column creation.

`-powerDomain` *`powerDomainName`*

Specifies the power domain that shuts off through the inserted power switches.

Note: This parameter is required for both power switch ring and column creation.

`-protoReportFile` *`fileName`*

Specifies the file containing prototyping results.

`-prototypeChainDepth {0|1}`

Performs prototyping on chain depth given maximum ramp up current. You must specify `-rampUpTime` when you specify this parameter.

`-prototypeChainDepthGivenRampUpCurrent {0|1}`

Performs prototyping on chain depth given maximum ramp up current. You must specify `-maxRampUpCurrent` when you specify this parameter.

`-prototypeDelayGivenRampUpCurrent {0|1}`

Analyzes optimum ramp up time given maximum ramp up current. You must specify `-maxRampUpCurrent` when you use this command.

`-prototypeIgnoreLeakage {0|1}`

Ignores leakage constraints during prototyping.

`-prototypeMaxChainDepth {0|1}`

Performs maximum chain depth prototyping based on ramp up time. You must specify `-rampUpTime` when you specify this parameter.

`-prototypeMinChainDepth {0|1}`

Performs minimum chain depth prototyping based on ramp up time. You must specify `-rampUpTime` when you specify this parameter.

`-prototypeNumberSwitches {0|1}`

Determines the optimum number of switches based on switch characteristics and domain constraints.

`-prototypeRampUpTime {0|1}`

Performs prototyping on ramp up time based on chain depth. You must specify `-rampUpChainDepth` when you specify this parameter.

`-prototypeSweepChainDepth {`*`min_integer max_integer increment_integer`*`}`

Explores various chain depths from minimum to maximum in specified increments, and reports ramp up metrics for each setting.

`-prototypeSweepSwitchNumber {`*`min_integer max_integer increment_integer`*`}`

Explores the number of switches from minimum to maximum in specified increments, and reports corresponding power domain operation metrics.

`-r0Corner` *`corner_syntax`*

Specifies the side where the corner cell instance can be placed in the `R0` orientation without rotation.

`UL` = Upper left
`LL` = Lower left
`LR` = Lower right
`UR` = Upper right

The orientation rotates the corner cell 90 degrees for each corner, proceeding counterclockwise.

Note: This parameter applies to power switch ring creation only.

Default: `UL`
The corner cell in the upper left position in the ring is placed with orientation R0.

Default orientations:

```
UL = R0
LL = R90
LR = R180
UR = R270
```

`-r0InsideCorner {LT | TR | RB | BL}`

Specifies the inside corner where to place the corner cell instance in the R0 orientation. Use this parameter for rectilinear switch rings containing inside corners.

`LT` = Corner of the left and top sides

`TR` = Corner of the top and right sides

`RB` = Corner of the right and bottom sides

`BL` = Corner of the bottom and left sides

Note: This parameter applies to power switch ring creation only.

Default: `LT`

The corner cell in the left top position in the ring is placed with the orientation R0.

```
LT = R0
BL = R90
RB = R180
TR = R270
```

`-r0Side {L | T | R | B}`

Specifies the side where the switch or filler instance can be placed in the `R0` orientation without rotation. The command adjusts the orientation of the remaining sides so that the switches form a continuous ring. The orientation rotates 90 degrees for each side, proceeding counterclockwise.

`L` = Left side
`T` = Top side
`R` = Right side
`B` = Bottom side

The orientation rotates 90 degrees for each side, proceeding counterclockwise, unless `-counterClockwise` is specified.

Note: This parameter applies to power switch ring creation only.

Default: `T`
The switch and filler cells in the top row are placed with orientation `R0`.

Default orientations:

```
T = R0
L = R90
B = R180
R = R270
```

`-r0SideOrientation` *`orientation_syntax`*

Enables the software to rotate or mirror the switch instances on the R0 side. The software automatically adjusts the orientation of the other sides of the domain to form a continuous ring.

Note: This parameter applies to power switch ring creation only.

`-rampUpChainDepth` *`integer`*

Specifies the chain depth used in ramp up time prototyping. Specify this parameter when you specify `-prototypeRampUpTime`.

`-rampUpRailVoltagePercent` *`float`*

Specifies the percentage of the full domain voltage upon which the power domain is considered operable when ramping up.

Default: 90 percent.

`-rampUpTime {`*`min_value_in_seconds max_value_in_seconds`*`}`

Specifies the minimum and maximum ramp up time settings for chain depth prototyping. This parameter is required for `-prototypeChainDepth`, `-prototypeMinChainDepth`, and `-prototypeMaxChainDepth`.

`-readPowerSwitchCell` *`fileName`*

Specifies the file containing the switch characterization data. The file contains a list of characterization data per cell, on one line. If you do not put all of the information for a cell on one line, the tool issues an error message. The format is as follows:

```
CELL cell_name SUPPLY pin_name SWITCHED pin_name RON
value_in_ohms IDSAT value_in_mA ILEAK value_in_mA
CELL2...
```

For power switch prototyping, the tool only reads and uses the first line of the file and uses only one cell type.

`-reportFile` *`filename`*

Writes switch instances and their corresponding input/output enable signal pin/net connections to the specified file name.

`-ring` Specifies ring style power switch insertion. Power switches are placed outside the power domain.

`-rightNumSwitch` *`integer`*

Specifies the number of switches to insert on right side(s) of the power domain.

Note: This parameter applies to power switch ring creation only.

Three scenarios can occur:

- If the spaces available match the number specified, then the software fills these spaces with switches.
- If the number exceeds the spaces available, the software issues a warning message.
- If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space.

Also, if you specify `f` instead of an integer, the tool automatically fills the right side with switches and filler cells.

This parameter is optional. See `-bottomNumSwitch`, `-leftNumSwitch`, and `-topNumSwitch`.

Default: `f`

`-rightOffset` Specifies the distance between the right power domain boundary and the nearest switch cell. This parameter can be used in either of two ways:

- Columns

  Specifies the distance between the right-most switch in the switch column and the right fence boundary of the power domain. You can use this parameter along with `-width`, `-height`, and `-leftOffset` to define a unique, specific area within the power domain for placing switches. By default, the switch column can abut the right fence boundary.

- Rings

  Specifies distance from the right edge outside of the power domain fence and the right side of the power switch ring. See `-horizontalOffset, -verticalOffset, -leftOffset`, `-bottomOffset`, and `-topOffset`.

  Note: When you create a power domain, a `minGap` exists for this power domain, and the switches are inserted within this `minGap`. However, if the specified `minGap` is less than the switch dimension, then the tool automatically extends the `minGap` for this power domain.

  For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

`-rightOrientation` *`orientation_syntax`*

Specifies the cell orientation for all right side(s) of the power switch ring.

Note: This parameter applies to power ring creation only.

`-rightPattern` *`pattern_syntax`*

Defines a pattern for the right side(s) of the ring only. You must use this parameter with `-bottomPattern`, `-topPattern`, and `-leftPattern`.

Note: This parameter applies to power switch ring only.

`-rightSide [0 | 1]`

Inserts switches on the right side(s) of the power domain. If you do not specify a side, the software creates a full power switch ring around the specified power domain. If you specify a side, the software inserts switches on that side. In this case, the tool sets `-rightSide` to `1` and sets other sides to `0`.

You can specify multiple sides and the software inserts switches only on the sides you specify. The tool sets the specified sides to `1` and sets the unspecified sides to `0`.

Is it true that you can specify multiple sides?

Note: This parameter applies to power switch ring creation only.

Default: `1`

`-rOn` *`value_in_ohms`*

Specifies the resistance when the switch is operated in linear mode. If the switch is a mother-daughter type (with two enables), you can specify two values corresponding to the stage-1 and stage2 enables respectively.

`-skipRows` *`integer`*

Specifies the number of rows to skip when placing a switch in a column. For example, `-skipRows 2` skips two rows before placing the next switch in a column. You can use this parameter with or without the `-checkerboard` parameter.

Note: This parameter applies to power switch column creation only.

`-sideEndOffsetSideList {`*`value ...`*`}`

Places the last switch at the specified distance from the end of the sides. Use this parameter with rectilinear power domains to specify a value for each side.

For more information, see “Using Pitch Control and Offsets” in the “Low Power Design” chapter of the *Encounter User Guide*.

Default: `0`

`-sideNumSwitchList {`*`value`* `…}`

Lists the number of switches to insert on each side. Each value in the list corresponds to a side specified with `-specifySideList`.

The following example shows which sides of an eight-sided power domain can have switches:

```
-specifySideList {1 0 1 0 1 0 1 0}
```

Ring switches can be placed at sides 1, 3, 5, and 7.

The following example shows that 20 switches can be placed on side 1, 20 on side 3, 10 on side 5, and 10 on side 7:

```
-sideNumSwitchList {20 0 20 0 10 0 10 0}
```

You must include a space between each item in the list.

You *must* specify a valid cell name for each side, even if no cells can be specified for a side.

The tools checks that the number of cells you specify is the same number you specified with `-specifySideList`, and issues an error message if it finds a discrepancy.

`-sideOffsetList {`*`value`* `…}`

Lists the offset values representing the distance between the power domain fence boundary and each side of the power switch ring. Each value in the list corresponds to a side specified with `-specifySideList`.

The following example show that all sides of an eight-sided power domain can have switches:

```
-specifySideList {1 1 1 1 1 1 1 1}
```

The following example shows an offset of 8 μm from side1, 8 μm from side2, 8 μm from side 3, 6 μm from side 4, 3 μm from side 5, 3 μm from side 6, 3 μm from side 7, and 6 μm from side 8.

```
-sideOffsetList {8 8 8 6 3 3 3 6}
```

Note: When you create a power domain, a `minGap` exists for this power domain, and the switches are inserted within this `minGap`. However, if the specified `minGap` is less than the switch dimension, then the tool automatically extends the `minGap` for this power domain.

Important

You must include a space between each item in the list.

You *must* specify a valid cell name for each side, even if no cells can be specified for a side.

The tools checks that the number of cells you specify is the same number you specified with `-specifySideList`, and issues an error message if it finds a discrepancy.

For more information, see “Using Pitch Control and Offsets” in the “Low Power Design” chapter of the *Encounter User Guide*.

Default: `0`
The switches abut the power domain fence if possible.

`-sideOrientationList {`*`orientation_syntax`* `…}`

Lists the orientations for cells in each side of the power switch ring. Each orientation in the list corresponds to a side specified

The following example shows which sides of an eight-sided power domain can have switches:

```
-specifySideList {1 1 1 1 1 1 1 1 }
```

Ring switches can be placed on any side.

The following example shows that cells in side 1 have orientation R90, R180 in side 3, R270 in side5, and R0 in side 7:

```
-sideOrientationList {R90 R90 R180 R180 R270 R270 R0 R0}
```

You must include a space between each item in the list.

You *must* specify a valid cell name for each side, even if no cells can be specified for a side.

The tools checks that the number of cells you specify is the same number you specified with `-specifySideList`, and issues an error message if it finds a discrepancy.

Note: This parameter apples to power switch ring creation only.

`-sidePatternList {`*`pattern_syntax`* `…}`

Specifies the cell pattern per side of a power switch ring. Each pattern in the list corresponds to a side specified with `-specifySideList`.

The following example shows which sides of an eight-sided power domain can have switches:

```
-specifySideList {1 0 1 0 1 0 1 0}
```

Ring switches can be placed at sides 1, 3, 5, and 7.

The following example shows that cells in side 1 have orientation R90, R180 in side 3, R270 in side5, and R0 in side 7:

```
-sidePatternList {{cell1 cell2 {cell3 * 6}} {cell1 cell2 {cell3
* 6}} {cell2 cell3 {cell4 * 6}} {cell2 cell3 {cell4 * 6}} {cell1
cell2 {cell3 * 6}} {cell1 cell2 {cell3 * 6}} {cell2 cell3 {cell4
* 6}}{cell2 cell3 {cell4 * 6}}
```

You must include a space between each item in the list.

You must include a valid pattern for each side, regardless of whether the side has any switches.

The tools checks that the number of cells you specify is the same number you specified with `-specifySideList`, and issues an error message if it finds a discrepancy.

Note: This parameter applies to power switch ring creation only.

`-startOffset` *`value`*

Places the first cell on every side of the power domain at the specified distance from the beginning of each side.

The beginning of each side is not necessarily determined by the power domain boundary. You can use `-leftOffset`, `-rightOffset`, `-topOffset`, `-bottomOffset`, `-horizontalOffset`, and/or `-verticalOffset` to determine the beginning of each side relative to the power domain boundary.

For more information, see “Using Pitch Control and Offsets” in the “Low Power Design” chapter of the *Encounter User Guide*.

Default: 0

Note: This parameter applies to power switch ring creation only.

`-sideStartOffsetList {`*`value ...`*`}`

Places the first cell at the specified distance from the starting location of the sides. Use this parameter with rectilinear power domains to specify a value for each side.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

Note: This parameter applies to power switch ring creation only.

`-snapToNearest`

Snaps the cell placement to the nearest legal location. This parameter can be used when a hard macro blocks part of a column. The switches that would overlap the hard macro blockage are placed at legal locations nearest to the expected original location.

You must specify either `-skipRows` or `-checkerboard` in order to create enough space for the cells to snap. If you specify `-skipRows`, you must set the value greater than zero.

Note: This parameter applies to power switch column creation only.

`-spacerCellName` *`cell_syntax`*

Specifies the spacer cells to place on either side of power switch cells within a power domain to prevent the tool from placing standard cells next to the switch cells.

Note: This parameter applies to power switch column creation only.

`-specifySideList {0|1 …}`

Specifies the sides of a power switch ring that can and cannot contain ring switches. In the list, 1 indicates that the side can contain ring switches, and 0 indicates that the side cannot contain ring switches. The list must contain either 0 or 1 for each side of the shape. The first member of the list indicates the first side relative to Corner 0, which is the lower left corner of the ring. If `counterclockwise` is specified, the first side is counterclockwise from Corner 0.

The following example shows which sides of an eight-sided power domain can have sw itches:

```
-specifySideList {1 0 1 0 1 0 1 0}
```

Ring switches can be inserted at sides 1, 3, 5, and 7.

You must include a space between each item in the list.

Note: This parameter applies to power switch ring creation only.

`-startEnableChainAtCorner` *`integer`* or *`corner_syntax`*

Specifies the corner where the switch ring insertion begins. This parameter can accept corner syntax or an integer, which accommodates more than four corners in a rectilinear power domain. The corner numbering starts with corner 0 at the lower left corner, and proceeds clockwise around the switch ring (unless `-counterclockwise` is specified). For example, `-startEnableChainAtCorner 1` starts the enable chain at the first corner found in a clockwise direction from corner 0.

Note: This parameter applies to power switch ring creation only.

`-startOffsetBottom` *`value`*

Places the first cell on the bottom side of the power domain at the specified distance from the beginning location of the side.

The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-leftOffset`, `-rightOffset` or `-verticalOffset` to determine the beginning location of the side relative to the power domain boundary.

Ring style:

- If the value is positive, the software places the cell to the left of the starting location, or to the right of the starting location if `-counterclockwise` is specified.
- If the value is negative, the software places the cell to the right of the starting location, or to the left of the starting location if `-counterclockwise` is specified.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

Note: This parameter applies to power switch ring creation only.

`-startOffsetHorizontal` *`value`*

Places the first cell on the horizontal side of the power domain at the specified distance from the beginning location of the sides.

The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-leftOffset`, `-rightOffset` or `-verticalOffset` to determine the beginning location of the side relative to the power domain boundary.

Ring style:

- If the value is positive, the software places the cell to the left of the starting location, or to the right of the starting location if `-counterclockwise` is specified.
- If the value is negative, the software places the cell to the right of the starting location, or to the left of the starting location if `-counterclockwise` is specified.

For more information, see “Using Pitch Control and Offsets” in the “Low Power Design” chapter of the *Encounter User Guide*.

Default: `0`

Note: This parameter applies to power switch ring creation only.

`-startOffsetLeft` *`value`*

Places the first cell on the left side of the power domain at the specified distance from the beginning location of the side.

The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-bottomOffset`, `-topOffset`, or `-horizontalOffset` to determine the beginning location of the side relative to the power domain boundary.

Ring style:

- If the value is positive, the software places the cell above the starting location, or below the starting location if `-counterclockwise` is specified.
- If the value is negative, the software places the cell below the starting location, or above the starting location if `-counterclockwise` is specified.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

Note: This parameter applies to power switch ring creation only.

`-startOffsetRight` *`value`*

Places the first cell on the right side of the power domain at the specified distance from the beginning location of the side.

The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-topOffset`, `-bottomOffset`, or `-horizontalOffset` to determine the beginning location of the side relative to the power domain boundary.

Ring style:

- If the value is positive, the software places the first cell below the beginning location of the side, or above the beginning location of the side if `-counterclockwise` is specified.
- If the value is negative, the software places the first cell above the beginning location of the side, or below the beginning location of the side if `-counterclockwise` is specified.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

Note: This parameter applies to power switch ring creation only.

`-startOffsetTop` *`value`*

Places the first cell at the specified distance from the starting location of the top offset.

Use this parameter with `-leftOffset`, `-rightOffset`, or `-verticalOffset` to determine the beginning location of the side.

Ring style:

- If the value is positive, the software places the first cell to the right of the beginning location of the side, or to the left of the beginning location of the side if `-counterclockwise` is specified.
- If the value is negative, the software places the first cell to the left of the beginning location of the side, or to the right of the beginning location of the side if `-counterclockwise` is specified.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

Note: This parameter applies to power switch ring creation only.

`-startOffsetVertical` *`value`*

Places the first cell on the vertical sides of the power switch ring at the specified distance from beginning location of the vertical sides.

The beginning location of the side is not necessarily determined by the power domain boundary. You can use `-topOffset`, `-bottomOffset`, or `-horizontalOffset` to determine the beginning location of the side relative to the power domain boundary.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

Note: This parameter applies to power switch ring creation only.

`-switchCellNameBottom` *`cell_syntax`*

Specifies a different switch cell for the bottom side(s) of the power switch ring.

Note: This parameter applies to power switch ring creation only.

Default: The switch cell specified by `-globalSwitchCellName.`

`-switchCellNameHorizontal` *`cell_syntax`*

Specifies a different switch cell for the horizontal side(s) of a power switch ring.

Note: This parameter applies to power switch ring creation only.

Default: The switch cell specified by `-globalSwitchCellName`.

`-switchCellNameLeft` *`cell_syntax`*

Specifies a different switch cell for the left side(s) of the power switch ring.

Note: This parameter applies to power switch ring creation only.

Default: The switch cell specified by `-globalSwitchCellName`.

`-switchCellNameRight` *`cell_syntax`*

Specifies a different switch cell for the right side(s) of the power switch ring.

Note: This parameter applies to power switch ring creation only.

Default: The switch cell specified by `-globalSwitchCellName`.

`-switchCellNameTop` *`cell_syntax`*

Specifies a different switch cell for the top side(s) of the power switch ring.

Note: This parameter applies to power switch ring creation only.

Default: The switch cell specified by `-globalSwitchCellName`.

`-swithCellNameVertical` *`cell_syntax`*

Specifies a different switch cell for the vertical side(s) of a power switch ring.

Note: This parameter applies to power switch ring creation only.

Default: The switch cell specified by `-globalSwitchCellName`.

`-switchModuleInstance` *`instanceName`*

Specifies module instance where all the switches will be instantiated. This parameter is required.

Note: This parameter applies to both power switch ring and column creation.

`-switchCellSidesList {`*`cell_syntax`* `…}`

Specifies the switch cells to insert on each side of a power domain. The first member of the list indicates the first side relative to Corner 0, which is the lower left corner of the ring. If `-counterclockwise` is specified, the first side is counterclockwise from Corner 0.

The following example shows the switch cells specified for an eight-sided power switch ring:

```
-specifySideList {1 0 1 0 1 0 1 0}
-switchCellSideList {cell1 cell1 cell2 cell2 cell3 \ cell3 cell4
cell4}
```

`cell1` is used for the first side. You can then specify `cell1` again to provide a valid cell type, even though there can be no cells on the second side.

Important

You must include a space between each item in the list.

You *must* specify a valid cell name for each side, even if no cells can be specified for a side.

The tools checks that the number of cells you specify is the same number you specified with `-specifySideList`, and issues an error message if it finds a discrepancy.

Note: This parameter applies to power switch ring creation only.

`-switchPitch` Specifies the global switch pitch value. You cannot use this parameter with the `-switchPitch*` parameters.

For more information, see “Using Pitch Control and Offsets” in the “Low Power Design” chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch ring creation only.

Default: `0`

`-switchPitchBottom` *`value`*

Places the a switch the specified distance from the right edge of the previous cell on the bottom side of the power domain. If `-counterclockwise` is specified, the distance is measured from the left edge of the previous cell.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch ring creation only.

Default: `0`

`-switchPitchHorizontal` *`value`*

Places the a switch the specified distance from the left edge of the previous cell on the horizontal side of the power domain. If `-counterclockwise` is specified, the distance is measured from the right edge of the previous cell.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch ring creation only.

Default: `0`

`-switchPitchLeft` *`value`*

Places the a switch the specified distance from the bottom edge of the previous cell on the left side of the power domain. If `-counterclockwise` is specified, the distance is measured from the top edge of the previous cell.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch ring creation only.

Default: `0`

`-switchPitchRight` *`value`*

Places the a switch the specified distance from the top edge of the previous cell on the right side of the power domain. If `-counterclockwise` is specified, the distance is measured from the bottom edge of the previous cell.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch ring creation only.

Default: `0`

`-switchPitchTop` *`value`*

Places the a switch the specified distance from the left edge of the previous cell on the top side of the power domain. If `-counterclockwise` is specified, the distance is measured from the right edge of the previous cell. You can use this parameter with `-switchPitch*`, `-endOffset*`, and `-*Offset` parameters.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch ring creation only.

Default: `0`

`-switchPitchSideList {`*`value ...`*`}`

Places the a switch the specified distance from the edge of the previous cell on the sides specified in the list. Use this parameter for rectilinear power domains. You can use this parameter with `-switchPitch*`, `-endOffset*`, and `-*Offset` parameters.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch ring creation only.

Default: `0`

`-switchPitchVertical` *`value`*

Places the a switch the specified distance from the bottom edge of the previous cell on the vertical sides. If `-counterclockwise` is specified, the distance is measured from the top edge of the previous cell. You can use this parameter with `-switchPitch*`, `-endOffset*`, and `-*Offset` parameters.

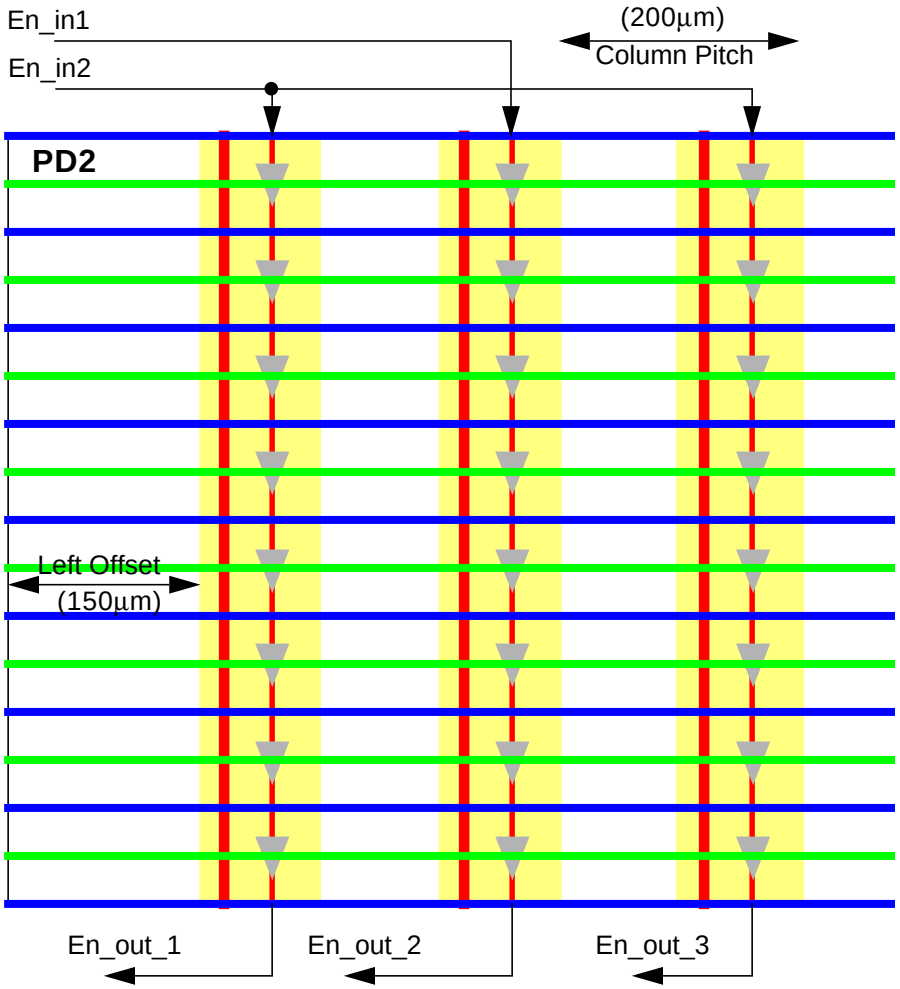
For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch ring creation only.

Default: `0`

`-topDown` Adds switches from the top down rather than from the bottom up.

The following figure shows how the software places switches from top to bottom.



Note: This parameter applies to power switch column creation only.

`-topOffset` *`float`*

Specifies the minimum offset from the top edge outside of the power domain fence boundary where the software can insert switches. See `-bottomOffset`, `-rightOffset`, and `-leftOffset`.

Note: This parameter applies to power switch ring creation only.

Note: When you create a power domain, a `minGap` exists for this power domain, and the switches are inserted within this `minGap`. However, if the specified `minGap` is less than the switch dimension, then the tool automatically extends the `minGap` for this power domain.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`
The switches abut the power domain fence if possible. Otherwise, the switches are placed as close as possible to the power domain.

`-topNumSwitch` *`integer`*

Specifies the number of switches to insert on the top side(s) of the power domain.

Note: This parameter applies to power switch ring creation only.

Three scenarios can occur:

- If the spaces available match the number specified, then the software fills these spaces with switches.
- If the number exceeds the spaces available, the software issues a warning message.
- If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space.

Also, if you specify `f` instead of an integer, the tool automatically fills the top side with switches and filler cells.

This parameter is optional. See `-horizontalNumSwitch`, `-verticalNumSwitch`, `-bottomNumSwitch`, `-leftNumSwitch`, and `-rightNumSwitch`.

Default: `f`

`-topOffset` *`value`*

Specifies the distance between the top-most switch in the switch column and the top fence boundary of the power domain that shuts off.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Note: This parameter applies to power switch column creation only.

Default: `0`

`-topOrientation` *`orientation_syntax`*

Specifies the cell orientation for all top side(s) of the power switch ring.

Note: This parameter applies to power ring creation only.

`-topPattern` *`pattern_syntax`*

Defines a pattern for the top side(s) of the ring only. You must use this parameter with `-horizontalPattern`, `-verticalPattern`, `-bottomPattern`, `-leftPattern`, and `-rightPattern.`

Note: This parameter applies to power switch ring only.

`-topSide [0 | 1]`

Inserts switches on the top side(s) of the power domain. If you do not specify a side, the software creates a full power switch ring around the specified power domain. If you specify a side, the software inserts switches on that side. In this case, the tool sets `-topSide` to `1` and sets the other sides to `0`.

You can specify multiple sides and the software inserts switches only on the sides you specify. The tool sets the specified sides to `1` and sets the unspecified sides to `0`.

Note: This parameter applies to power switch ring creation only.

Default: `1`

`-totalLeakagePower` *`value_in_watts`*

Specifies the total leakage power of the power domain before switches are added.

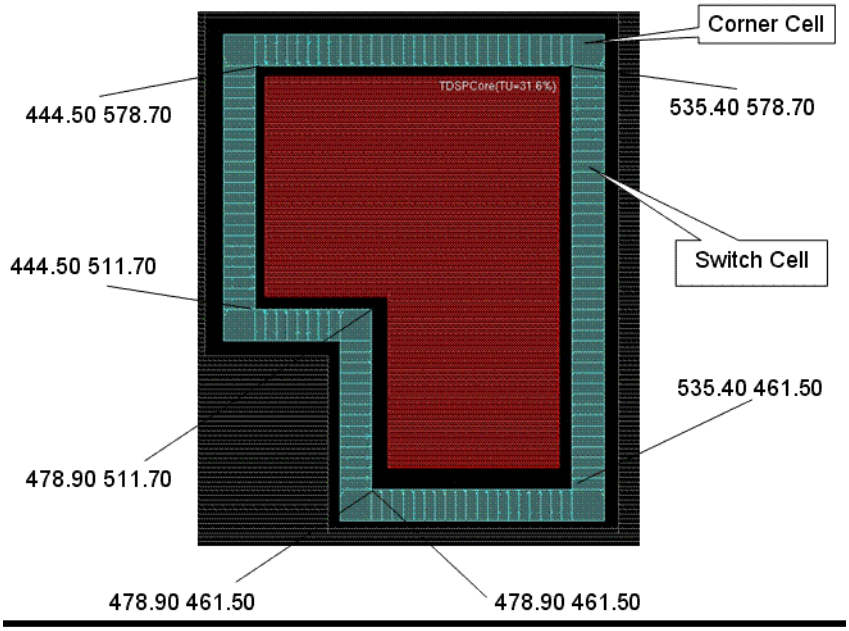
`-totalPower` *`value_in_watts`*

Specifies the total power consumed by the domain. If you do not specify this parameter, `report_power` computes power consumption.

`-vertex` *{x1 y1 x2 y1 x2 y2 x3 y2 x3 y3 x1 y3 ... xn yn}*

Allows you to insert power switches in ring style around the power domain by specifying vertex locations around the power domain. In the ring the cells are towards the outer side of the edge.

The `-vertex` parameter is used with the `-ring` parameter.




```
addPowerSwitch \
-ring -vertex \
{478.90 461.50 478.90 511.70 444.50 511.70 \
444.50 578.70 535.40 578.70 535.40 461.50 \
478.90 461.50} -powerDomain TDSPCore...
```

`-verticalNumSwitch` *`value`*

Specifies the number of switches to insert on all vertical side(s) of the power domain.

Note: This parameter applies to power switch ring creation only.

Three scenarios can occur:

- If the spaces available match the number specified, the software fills these spaces with switches.
- If the number exceeds the spaces available, the software issues the following error message.
- If the number is less than the spaces available, the tool inserts the switches, then inserts filler cells to fill the space.

Also, if you specify `f` instead of an integer, the tool automatically fills the bottom side with switches and filler cells.

This parameter is optional. See `-leftNumSwitch`, `-rightNumSwitch`, and `-topNumSwitch`.

Default: `f`

`-verticalOffset` *`value`*

Offsets the switch ring from the vertical edge of the power domain. Specify the offset in μm.

For more information, see "Using Pitch Control and Offsets" in the "Low Power Design" chapter of the *Encounter User Guide*.

Default: `0`

`-verticalOrientation` *`orientation_syntax`*

Specifies the cell orientation for all vertical side(s) of the power switch ring.

`-verticalPattern` *`pattern_syntax`*

Specifies a pattern of cells to repeat on all vertical side(s) of the power domain. The pattern consists of a group of different types of cells in a specified order.

Note: This parameter applies to power switch ring creation only.

`-verticalSide [0|1]`

Inserts ring switches on all vertical side(s) of a power domain.

Note: This parameter applies to power switch ring creation only.

Default: 1

`-voltage` *`value_in_volts`*

Specifies the primary rail voltage of the power domain.

`-width` *`length_in_um`*

Specifies the width of an area inside the power domain where switches can be placed. Specifies the distance between the bottom-most switch in the switch column and the bottom fence boundary of the power domain that shuts off. You can use this parameter along with `-bottomOffset`, `-height`, and `-leftOffset` to define a unique, specific area in the power domain for placing switches.

Note: This parameter applies to power switch column creation only.

Examples

- The following command creates a power switch column within power domain `PD`:

```
addPowerSwitch \
-column \
-powerDomain PD \
-globalSwitchCellName switchCell \
-switchModuleInstance switchBlock \
-enablePinIn A0 \
-enablePinOut Z0 \
-enableNetIn {enableNetOn} \
-enableNetOut {enableNetOut} \
-leftOffset 30 \
-bottomOffset 0 \
-horizontalPitch 240
```

  In this example, the software inserts `switchCell` into power domain `PD`. This switch cell has only one buffer inside. This example specifies one `-enableNetIn` (`enableNetOn`) and one `-enableNetOut` (`enableNetOut`). The input pin `A0` on the first switch of every column is driven by the net `enableNetOn`. The `enablePinOut` (`Z0`) drives the `enablePinIn` pin of the next switch cell in the same column. The nets for this connection are created internally and the names are unique. The `enablePinOut` pin of the last switch for each column is tied to the net `enableNetOut`. The first switch cell is placed 30 microns from the left side boundary of the selected power domain. The spacing between the columns is 240 microns.

- The following command creates a power switch ring around power domain `PD`.

```
addPowerSwitch \
-ring \
-powerDomain PD \
-switchModuleInstance switchBlock \
-enablePinIn {A0 A1} \
-enablePinOut {Z0 Z1} \
-enableNetIn enIn \
-enableNetOut enOut \
-globalSwitchCellName switchCell \
-globalFillerCellName fillerCell \
-firstInstanceName u1_switchCell \
right f \
-rightNumSwitch 28 \
-topNumSwitch 40 \
-bottomNumSwitch 50
```

In in this example, all `switchCell` switches are instantiated in the `switchBlock` module.

Note: The bottom offset is zero in this example. The first switch cell is placed in the column in or on the bottom-most switch cell row. If the row sites are different for switch cells, you can create them with the `createShifterRows -site` *`switch_cell_site`* command.

This example inserts the switches as follows:

- Switches are inserted in a full ring pattern (left, top, right, and bottom)
- The switch ring encircles the `PD` power domain fence. `PD` is the power domain that shuts off.
- A specified number of switches are inserted on each side of the ring. The `-numSwitchLeft` parameter has a value of `f` (autofill), so the tool fills the left side with as many switches as possible in the available space, then inserts filler cells.

  If the spaces available match the number of switches specified, then the software fills these spaces with switches. If the number of switches is less than the spaces available, the tool inserts the switches, then inserts filler cells `fillerCell` to fill the space. If the number exceeds the spaces available, then the software issues the following error message:

  ```
  ERROR: number of switches exceeded space availability.
  ```

  Also in this example, the original netlist already has a switch instantiated inside the module `switchBlock`. The switch is fixed as the first instance in the ring by `-firstInstanceName u1_switchCell`. The tool then generates a unique instance name for each switch.

  You must specify the input and output enable pin names of the switch. You must specify the pins in order because the software attaches them to the enable nets you specify. In this example, there is a loopback type of enable nets, so you must specify

one net for input (`-enableNetIn`) and one net for output (`-enableNetOut`). The input net (`enIn`) attaches to the first variable specified in the `-enablePinIn` list (`A0`) of the first switch, and the output net (`enOut`) attaches to the last variable specified in the `-enablePinOut` list (`Z1`) of the next switch inserted.

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"
  - "Power Shutdown Techniques"
  - "Power Switch Optimization"

addShifter

```
addShifter
     [-excNet netName]
     [-excNetFile fileName]
     [-noFTermShifter]
     [-forceShifterInPDHInst | -noForceShifterInPDHInst]
     [-prefix]
     [-selNet netName]
     [-selNetFile fileName]
```

Automatically inserts voltage level shifters into the location power domain (as defined in the shifter table) for signals that traverse power domains.

In addition to inserting voltage level shifters, this command also does the following:

- Identifies all the interface signals (and their direction) between power domains.
- For each signal that goes from a driving domain to a receiving domain that is defined in the *shifter_file_name* above, the corresponding *cell_name* is inserted in the location power domain. Voltage level shifters are not inserted for interface signals that go between power domains whose names are not defined in the *shifter_file_name*.
- Updates the connectivity in the database.

Note: The `addShifter` command issues an error message if no voltage level shifter is defined (the `loadShifter` command was not executed).

The `addShifter` command is obsolete. To ensure compatibility with future releases, create a Common Power Format (CPF) file and include the related CPF commands.

For backward compatibility, the `addShifter` command still works in this release, but the tool issues a warning message. By default, the design created with this command cannot be saved to the Encounter database. To save an Encounter database, set the following variable:

```
setVar dbgSaveOldMsvDbOnly 1
```

Parameters

`-excNet` *`netName`*

The specified net names are prevented from having a level shifter added. This allows you to explicitly exclude critical signals used for enabling power or clock in the power domain.

If you specify multiple net names, separate them with a space.

`-excNetFile` *`fileName`*

The list of net names in the specified file are to be excluded from having a level shifter added. The file format is a list of net names, for example:

```
net1
net2
net3 …
```

`-forceShifterInPDHInst | -noForceShifterInPDHInst`

Forces the software to add shifters cells to each HInst in the location power domain as defined in the shifter table. This is the default behavior. To suppress this behavior, use `-noForceShifterInPDHInst`.

Default: `-forceShifterInPDHInst`

`-noFTermShifter`

Prevents the software from adding level shifters to nets that cross power domains if the net is connected to an fterm.

Default: Adds a shifter to a net that traverses power domains and connects to or from an fterm. The software assumes that the fterm is located in the default power domain.

`-prefix` Adds the specified prefix to the added instance names. This prefix overrides the default prefix.

Default: `msvBuf_`

`-selNet` *`netName`* Adds level shifters to the specified nets.

`-selNetFile` *`fileName`*

Specifies the name of the file containing a list of nets for the software to include when adding level shifters. You must list each net on a separate line in the file.

Command Order

Use this command after the following condition is met:

- The voltage level shifters are identified by the command

  `loadShifter -infile` *`shifter_file_name`*

  For more information, see “loadShifter” on page 371.

adjustPowerDomainToAlignRows

```
adjustPowerDomainToAlignRows
     -refPD powerDomainName
     -powerDomain powerDomainName ...
```

Stretches the bounding box of a power domain so that its single-height rows align with those in a reference power domain, and the orientations match.

Note: You must first create rows in power domains.

Parameters

`-powerDomain` *`powerDomainName …`*

Specifies the power domain whose rows you want to align with a reference power domain.

`-refPD` *`powerDomainName`*

Specifies the reference power domain.

Examples

The following command stretches the bounding box of `PD2` so that the its single-height rows align with those in reference domain `PD1`:

```
adjustPowerDomainToAlignRows -refPD PD1 -powerDomain PD2
```

bufferTreeSynthesis

```
bufferTreeSynthesis
     -nets list_of_nets
     {-bufList list_of_buffers | -powerDomainBufList pd_buffer_list}
     [-fixedBuf]
     [-fixedNet]
     [-leafPorts port_list]
     [-maxDelay value]
     [-maxSkew value]
     [-maxTran value]
     [-minDelay value]
     [-noGating NO|Rising|Falling]
     [-prefix prefix_string]
     [-specifiedPowerDomainsOnly]
     [-srpgEnablePins]
```

Buffers high-fanout nets that are not clock nets. This command supports always-on buffer insertion for always-on nets, such as the enable nets of state-retention cells.

Parameters

`-bufList` *`list_of_buffers`*

Specifies the buffers to insert on the high-fanout nets.

`-fixedBuf` Sets the buffers added by this command as `dont_touch`.

`-fixedNet` Sets the nets added by this command as `dont_touch`.

`-leafPorts` *`port_list`*

Specifies the list of leaf ports. This parameter corresponds to the `LeafPort` value in the clock tree synthesis technology file. For more information, see `specifyClockTree`.

`-maxDelay` *`value`* Specifies the maximum phase delay constraint. This parameter corresponds to the `MaxDelay` value in the clock tree synthesis technology file. For more information, see `specifyClockTree`.

Default: `10` ns.

`-maxSkew` *`value`* Specifies the maximum skew constraint. This parameter corresponds to the `MaxSkew` value in the clock tree synthesis technology file. For more information, see `specifyClockTree`.

Default: `10` ns

`-maxTran` *`value`*

Specifies the maximum transition time constraint. This parameter corresponds to the `BufMaxTran` value in the clock tree synthesis technology file. For more information, see `specifyClockTree`.

Default: The `maxTran` value retrieved from the timing library.

`-minDelay` *`value`*

Specifies the minimum phase delay constraint.

`-nets` *`list_of_nets`*

Specifies which nets to buffer. This parameter is required.

`-noGating NO|rising|falling`

Sets the criteria for tracing through logic gates.

Choose only one of the following:

`rising`	Stops tracing through a gate (including buffers and inverters) and treats the gate as a rising-edge-triggered flip-flop clock pin.
`falling`	Stops tracing through a gate (including buffers and inverters) and treats the gate as a falling-edge-triggered flip-flop clock pin.
`NO`	Allows CTS to trace through clock gating logic.

Default: This command stops tracing through a gate.

`-powerDomainBufList` *`pd_buffer_list`*

Specifies the buffers to use for each specified power domain. The *`pd_buffer_list`* must have the following format:

```
{{pd1 buf1_1...} {pd2 buf2_2...}...}
```

`-prefix` *`prefix_string`*

Adds the specified prefix to the buffers. If *`prefix_string`* is specified, *`prefix_string_power_domain_name`* is used. Otherwise, FE_A0BUF*_power_domain_name* is used.

`-specifiedPowerDomainsOnly`

Only adds buffers in the power domains specified in `-powerDomainBufferList`.

`-srpgEnablePins` Automatically adds all the SRPG cells' enable pins to leaf ports.

Examples

The following command does buffer tree synthesis as follows:

- Buffers `enable0`
- Marks buffers as `dont_touch`
- Uses `BUFFD2BWP`, `BUFFD4BWP`, and `BUFFD8BWP` for power domain `A`; and `PTBUFFD2BWP` for `TDSPCore`

```
bufferTreeSynthesis -nets enable0 -fixedBuf -powerDomainBufList {{A) BUFFD2BWP
BUFFD4BWP BUFFD8BWP} {TDSPCore PTBUFFD2BWP}}
```

cleanRedundantShifter

```
cleanRedundantShifter
     [-excNet netNames]
     [-excNetFile fileName]
```

Delete back-to-back level shifters created by timing optimization in order to delete buffers.

Parameters

`-excNet` *`netNames`*

Retains back-to-back level shifters on the specified excluded nets.

`-excNetFile` *`fileName`*

Retains back-to-back level shifters on the nets named in the specified file.

Examples

The following command deletes back-to-back level shifters on all nets except `Net1`:

```
cleanRedundantShifters -excNet Net1
```

commitCPF

```
commitCPF
     [-keepRows]
     [-powerDomain]
     [-verbose]
```

Runs the commands provided in the loaded CPF file, such as `create_power_domain`, `create_power_ground_connection`, and `create_*_logic`. As a result, running this command does the following:

- Creates power domains
- Creates logical power/ground net connections
- Specifies timing libraries for power domains
- Inserts shifter cells and isolation cells
- Replaces regular registers with state-retention (SRPG) registers

After running this command, you can then place and floorplan the power domains.

Parameters

`-keepRows`	Retains rows defined in the DEF file imported into Encounter. By default, the command removes rows.
`-powerDomain`	Commits power domains only, and does not commit isolation, level shifter, and state retention CPF commands.
`-verbose`	Prints to the logfile the native Encounter commands that `commitCPF` runs.

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Support for the Common Power Format"
- CPF Script Example chapter of the *Encounter User Guide*

createPowerDomain

```
CreatePowerDomain
     name
     {-maxTimingLibs timing_lib_list
     -minTimingLibs timing_lib_list
     [[-commonTimingLibs timing_lib_list] |
     [-timingLibs timing_lib_list] |
     [-default]]}
     [-minGaps T B L R]
     [-rsExts T B L R]
     [-rowFlip {first | second | noFlip | auto}]
     [-rowSpaceType {1 | 2}]
     [-rowSpacing value]
     [-alwaysOn true|false]
```

Defines a power domain.

Use this command after you complete floorplanning for your design.

Note: You *must* create a default power domain after you create non-default power domains.

Important

The `createPowerDomain` command is obsolete. To ensure compatibility with future releases, create a Common Power Format (CPF) file and include the related CPF commands.

For backward compatibility, the `createPowerDomain` command still works in this release, but the tool issues a warning message. By default, the design created with this command cannot be saved to the Encounter database. To save an Encounter database, set the following variable:

```
setVar dbgSaveOldMsvDbOnly 1
```

Parameters

`-alwaysOn true | false`

Specifies whether a power domain is always on.

Default: `true`. The power domain is always on.

`-commonTimingLibs` *`timing_lib_list`*

Specifies the names of the common timing libraries associated with the power domain. All the cells and macros in the power domain obtain their timing and power information from these libraries. You can specify either TLF or `.lib` timing libraries. This feature is not supported in the Create Power Domains GUI, which invokes the `-timingLibs` option.

TLF timing libraries are supported for backward compatibility only. Cadence recommends that you use `.lib` libraries.

Note: You must enclose the timing library list in braces, for example, `{lib1 lib2}`

You cannot use this parameter to bind timing libraries to power domains when the software is in multi-mode multi-corner mode. Instead, use the `update_delay_corner` command.

`-default` Creates a default power domain to enclose the non-default power domains and unassigned instances.

`-maxTimingLibs` *`timing_lib_list`*

Specifies the names of the max timing libraries associated with the power domain. All the cells and macros in the power domain obtain their timing and power information from these libraries. You can specify either TLF or `.lib` timing libraries. This feature is not supported in the Create Power Domains GUI, which invokes the `-timingLibs` option.

Note: You must enclose the timing library list in braces, for example, `{lib1 lib2}`

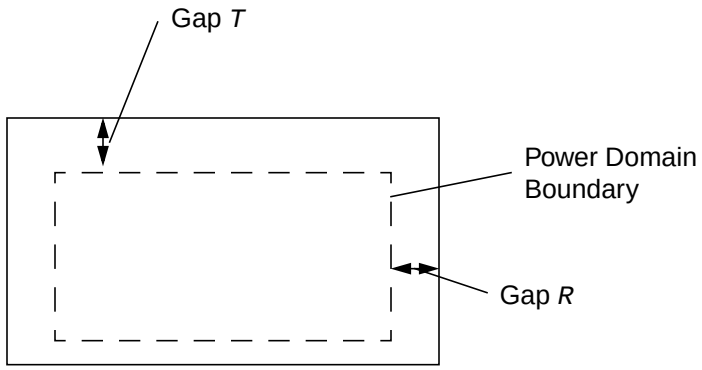
You cannot use this parameter to bind timing libraries to power domains when the software is in multi-mode multi-corner mode. Instead, use the `update_delay_corner` command.

`-minGaps` *`T B L R`*

Defines the distance, in microns, that must be reserved outside the power domain boundary edges for power routing. If you use this parameter, you must specify four values, one for each edge:

B	Specifies the distance reserved from the bottom edge.
L	Specifies the distance reserved from the left edge.
R	Specifies the distance reserved from the right edge.
T	Specifies the distance reserved from the top edge.

If you do not use this parameter, no gap is reserved around the power domain boundary edges. The following illustration shows the location of the gap in relation to the power domain.



`-minTimingLibs` *`timing_lib_list`*

Specifies the names of the min timing libraries associated with the power domain. All the cells and macros in the power domain obtain their timing and power information from these libraries. You can specify either TLF or `.lib` timing libraries. This feature is not supported in the Create Power Domains GUI, which invokes the `-timingLibs` option.

Note: You must enclose the timing library list in braces, for example, `{lib1 lib2}`

You cannot use this parameter to bind timing libraries to power domains when the software is in multi-mode multi-corner mode. Instead, use the `update_delay_corner` command.

`name`

Specifies the name of the power domain you want to create. You must specify this value immediately after the `createPowerDomain` command name.

`-rowFlip {first | second | noFlip | auto}`

Flips the orientation of either the first or second row, then follows the flipped and abutted row pattern. The `noFlip` option preserves the `R0` orientation for all rows.

The software automatically matches the orientation for rows inside the power domain with the orientation for rows outside the power domain. The `auto` option resets the software for this automatic (default) behavior if you have previously specified `first`, `second`, or `noFlip`.

Default: The software matches the row orientation inside and outside the power domain.

`-rowSpaceType {1 | 2}`

Determines whether the row spacing value applies between each row (1) or each pair of rows (2).

`-rowSpacing` *`value`*

Specifies the spacing between each row or pair of rows, depending on the `-rowSpaceType` setting.

The value can be any floating point number.

`-rsExts`

Limits the search for legal targets by the power planning and power routing software to a specified distance beyond the power domain boundary. This prevents unwanted connections to targets of the same net that are not associated with the power domain. If you use this parameter, you must specify four values, one for each edge:

B — Specifies the distance from the bottom edge of the power domain in which targets can be connected.

L — Specifies the distance from the left edge of the power domain in which targets can be connected.

R — Specifies the distance from the right edge of the power domain in which targets can be connected.

T — Specifies the distance from the top edge of the power domain in which targets can be connected.

Default: If you do not specify this parameter, only targets within the power domain boundary are used.

`-timingLibs` *`timing_lib_list`*

Specifies the names of the timing libraries associated with the power domain. All the cells and macros in the power domain get their timing and power information from these libraries. You can specify either TLF or `.lib` timing libraries.

You cannot use this parameter to bind timing libraries to power domains when the software is in multi-mode multi-corner mode. Instead, use the `update delay corner` command.

Examples

The following command creates a power domain, TDSP, with max timing libraries `hvt_worst` and `sclp_worst` and min timing libraries `hvt_best` and `sclp_best`.

```
createPowerDomain TDSP -maxTiminglibs {hvt_worst sclp_worst} -minTiminglibs
{hvt_best sclp_best} -minGaps {0.14} {0.14} {70} {0.14} -rsExts {50} {50} {70} {50}
```

createPowerDomainCut

`createPowerDomainCut` *`llx lly urx ury`*

The `createPowerDomainCut` command allows you to specify a rectilinear power domain by creating a new boundary for an existing power domain. This is done by specifying the coordinates of areas to be removed from the power domain.

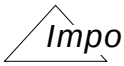
Parameters

`llx`	Specifies the lower left x coordinate for the area to be cut from the power domain.
`lly`	Specifies the lower left y coordinate for the area to be cut from the power domain.
`urx`	Specifies the upper right x coordinate for the area to be cut from the power domain.
`ury`	Specifies the upper right y coordinate for the area to be cut from the power domain.

createPowerMode

```
createPowerMode
     name
     {-onDomains power_domain_list |
     -offDomains power_domain_list}
```

Checks for shifter table completeness when you load the shifter table, based on the rules determined by the parameters in this command.



Important

The `createPowerMode` command is obsolete. To ensure compatibility with future releases, create a Common Power Format (CPF) file and include the related CPF commands.

For backward compatibility, the `createPowerMode` command still works in this release, but the tool issues a warning message. By default, the design created with this command cannot be saved to the Encounter database. To save an Encounter database, set the following variable:

```
setVar dbgSaveOldMsvDbOnly 1
```

Parameters

name
: Specifies the name of the power mode you create when you specify `-offDomains` or `-onDomains`. You can create as many power modes as you need.

`-offDomains` *power_domain_list*
: Specifies a list of power domains that are considered off in the specified power mode. If a power domain is not specified in the power domain list, the software assumes that it is on.

`-onDomains` *power_domain_list*
: Specifies a list of power domains that are considered on in the specified power mode. If a power domain is not specified in the power domain list, the software assumes that it is off.

Examples

- For power domains `PD1`, `PD2`, `PD3`, and `PD4`, the following command creates power mode `poweroffmode`:

```
createPowerMode poweroffmode -offDomains {PD1 PD2}
```

The PD1 and PD2 power domains are designated as off. PD3 and PD4, which are not specified in this command, are assumed to be on.

- For power domains PD1, PD2, PD3, PD4, mydomain, and anotherdomain, the following command creates power mode poweronmode:

  ```
  createPowerMode poweronmode {mydomain anotherdomain}
  ```

  The mydomain and anotherdomain power domains are designated as on. PD1, PD2, PD3, and PD4, which are not specified in this command, are assumed to be off.

Related Topics

- CPF Script Example chapter of the *Encounter User Guide*

createShifterRows

```
createShifterRows
     -powerdomain powerDomainName
     -site siteName
     [-depth integer [-edge]]
     [-flip1st]
     [-noFlip]
     [-edge {left | right}
```

Creates rows specifically for shifters in a power domain. You do not need to specify the power domain bounding box.

You must first create power domains.

Parameters

`-depth` *`integer`*

Specifies the number of instantiated sites for each core row.

The parameter applies to composite rows. For more information about composite rows, see the *LEF/DEF Language Reference*.

`-edge {left | right}`

Creates the rows only on the left side or only on the right side of the power domain. You must specify `-depth` when you use this parameter.

`-flip1st` Flips the orientation of the first shifter row. Follows the flipped and abutted row pattern unless `-noFlip` is specified.

`-noFlip` Does not flip alternate rows: All the rows have the same orientation.

`-powerDomain` *`powerDomainName`*

Specifies the power domain in which you want to create the shifter rows. You can create rows for the default power domain in addition to creating rows for the non-default power domains.

`-site` *`siteName`* Specifies the site for the rows.

Examples

The following commands delete all rows, create shifter rows for `PD1` using single-height site `singlesite1`, create shifter rows for `PD2` using double-height site `doublesite`:

```
createShifterRows -powerdomain PD1 -site singlesite1
createShifterRows -powerdomain PD2 -site doublesite2
loadShifter -infile MyFile
addShifter ...
placeDesign ...
```

cutPowerDomainByOverlaps

`cutPowerDomainByOverlaps` *`powerDomainName`*

Enables a power domain to contain a nested power domain. You must first create the power domains, then modify the power domain attributes to specify a bounding box for each domain. The bounding box for the inner domain must lie completely within the bounding box for the outer power domain. Instances in the outer power domain are not placed within the inner power domain.

Parameters

`powerDomainName`	Specifies the outer power domain where the nested power domain will reside.

Examples

To create nested power domains, create two power domains and specify the bounding box for each domain:

```
createPowerDomain PD1 ...
modifyPowerDomainAttr PD1 -box llx lly urx usy
createPowerDomain PD2 ...
modifyPowerDomainAttr PD2 -box llx lly urs usy
```

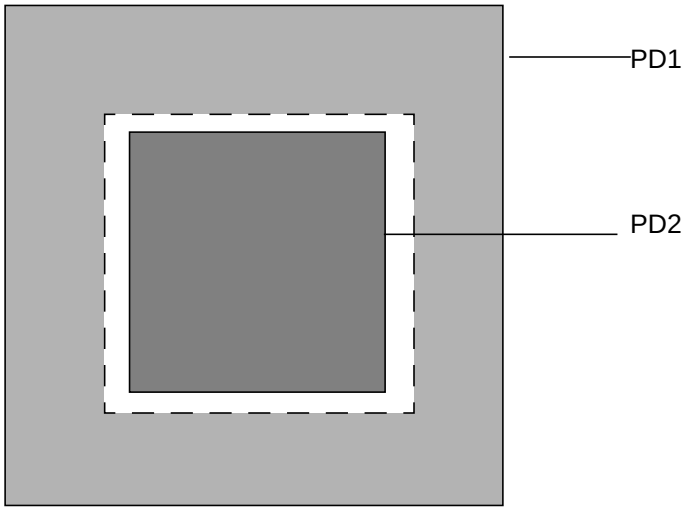
Note: The bounding box for `PD2` must lie completely within the `PD1` bounding box.

Designate `PD1` as the hollow power domain:

```
cutPowerDomainByOverlaps PD1
```

Instances in `PD1` will not be placed within `PD2`.

The following figure shows the resulting power domains.



deletePowerDomain

```
deletePowerDomain
     powerDomainName
```

Removes specified or all power domains.

Parameters

powerDomainName

Removes the specified power domains.

Default: The software removes all power domains.

Examples

The following command removes PD1 and PD2:

```
deletePowerDomain PD1 PD2
```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"

deletePowerSwitch

```
deletePowerSwitch
     {{-column | -ring}
     -powerDomain powerDomainName}
     [-instanceList {instName ...}]
```

Removes all switches and filler cells that were inserted by the `addPowerSwitch` command, from either columns or rings, or removes specified power switch.

This command retains and unplaces switches added by `optPowerSwitch -switchInstances`, whose cells were already instantiated in the netlist.

Parameters

`-column` — Removes all switches and filler cells that were inserted by the `addPowerSwitch` command, from columns in the specified power domain.

`-instanceList {`*`instName …`*`}` — Specifies the power switch cell instances to delete.

`-powerDomain` *`powerDomainName`* — Specifies the power domain next to which the columns are placed.

`-ring` — Removes all switches and filler cells that were inserted by the `addPowerSwitch` command, from the ring around the specified power domain.

Examples

- The following command removes instances `psoI_TDSP3_instA` and `psoI_TDSP3_InstB` that were added by the `addPowerSwitch` command:

  ```
  deletePowerSwitch -instanceList {psoI_TDSP3_instA psoI_TDSP3_InstB}
  ```

- The following command removes all power switch instances that were inserted by `addPowerSwitch` in columns inside the power domain `PD2`:

  ```
  deletePowerSwitch -column -powerDomain PD2
  ```

- The following command removes all power switch instances that were inserted by `addPowerSwitch` in the ring around `PD1`:

  ```
  deletePowerSwitch -ring -powerDomain PD1
  ```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Power Shutdown Techniques"

genShifterTable

```
genShifterTable
     fileName
```

Uses the data defined in the timing library to generate the shifter table data. The table lists all the possible shifter selections for the power domain pairs. You must manually edit the file to choose the correct shifter for each power domain pair.

You must first create power domains.

Important

The `genShifterTable` command is obsolete. To ensure compatibility with future releases, create a Common Power Format (CPF) file and include the related CPF commands.

For backward compatibility, the `genShifterTable` command still works in this release, but the tool issues a warning message. By default, the design created with this command cannot be saved to the Encounter database. To save an Encounter database, set the following variable:

```
setVar dbgSaveOldMsvDbOnly 1
```

Parameters

fileName	Specifies the name of the generated file.

Examples

You can use the following sequence of steps and create a shifter table `shifter.tbl`:

1. Load the config file
2. Load floorplan file
3. `createPowerdomain …`
4. `modifyPowerDomainMember …`
5. `modifyPowerDomainAttr …`
6. `genShifterTable shifter.tbl`

This command creates a default shifter table with the shifters, isolation cells based on the attributes in the new liberty syntax. You then must edit this table to provide information such as like the location of shifters.

getCPFUserAttributes

```
getCPFUserAttributes
     -domain domainName | -net netName
```

Reports the `user_attributes` of the power domain or net specified in the CPF file. If the domain or net has no `user_attributes` in the CPF file, then this command reports an empty string.

Parameters

`-domain`	Specifies the name of the power domain containing the user_attributes.
`-net`	Specifies the name of the net having the `user_attributes`.

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"
  - "Support for the Common Power Format"

getPowerDomainPwrGndPin

`getPowerDomainPwrGndPin` *`pd_name`*

Identifies all the power and ground pins of the instances in the specified power domain.

Parameters

`pdName`	Specifies the name of the power domain.

Examples

The following command outputs the power and ground pins for the instances in power domain PD3v:

```
getPowerDomainPwrGndPin PD3v
```

hilitePowerDomain

`hilitePowerDomain` *`powerDomainName`*

Highlights a power domain by placing a yellow box around all of its members.

Parameters

`powerDomainName`	Specifies the name of the power domain to be highlighted.

Examples

The following command highlights all members of `PowerDomain1`:

```
hilitePowerDomain PowerDomain1
```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"
  - "Highlight Power Domains"

loadCPF

`loadCPF` *`fileName`*

Reads the CPF file and performs lint, parsing, and semantics checking. The `loadCPF` command does not execute any of the commands within the CPF file.

If the power/ground connections are incomplete, then the `loadCPF` command automatically generates a power net specification template in CPF. You can then examine the template and update it to fully specify the power/ground connections. This command is equivalent to `globalNetConnect` in Encounter, but with limited power domain-oriented functionality.

The following example shows a template for power connections for PD1:

```
set VDD_PD1 “VDD_PD1”
create_power_ground_connection -pin VDD -net $VDD_PD1 -power_domain PD1
```

Parameters

`fileName`	Specifies the names of the files to load.

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - “Overview”
  - “Power Domain Shutdown and Scaling”
  - “Multiple Supply Voltage Flat Flow”
  - “Support for the Common Power Format”
- CPF Script Example chapter of the *Encounter User Guide*

loadShifter

```
loadShifter {-infile fileName | -template}
```

Identifies the cell names for all voltage level shifters and isolation cells used in the design netlist.

Note: You must first load the design.

Important

The `loadShifter` command is obsolete. To ensure compatibility with future releases, create a Common Power Format (CPF) file and include the related CPF commands.

For backward compatibility, the `loadShifter` command still works in this release, but the tool issues a warning message. By default, the design created with this command cannot be saved to the Encounter database. To save an Encounter database, set the following variable:

```
setVar dbgSaveOldMsvDbOnly 1
```

Parameters

`-infile` *`fileName`*
: Specifies the name of the file that contains information about voltage level shifters. For information on the template file format, see the Example below, or see the “Low Power Design” chapter of the *Encounter User Guide*.

`-template`
: Generates a template file named `template.vsf` that you can use for adding voltage level shifter information. For information on the template file format, see the “Low Power Design” chapter of the *Encounter User Guide*.

Examples

- The following command identifies `LevelShifterFile1` as the file that contains voltage level shifter information:

  ```
  loadShifter -infile LevelShifterFile1
  ```

modifyPowerDomainAttr

```
modifyPowerDomainAttr
     pdName
     [-box llx lly urx ury]
     [-cells {cell_list}]
     [-colFiller]
     [-isolation]
     [-minGaps T B L R]
     [-name newName]
     [-powerSwitch]
     [-rowFiller]
     [-rowFlip {first | second | noflip | auto}]
     [-rowSpacing value]
     [-rowSpaceType {1|2}]
     [-rsExts T B L R]
     [-shifter]
     [-side {topBottom rightLeft}]
     [-splitter]
     [-tap]
     [-wellSeparator]
```

Renames a power domain or changes its structure.

Parameters

`-box` *llx lly urx ury*

Specifies the coordinates within the design display area that serve as the boundary for the updated power domain.

`-cells {`*cell_list*`}`

Applies the power/ground connection rules to the specified instances. Use this parameters for level shifter, isolation, and power switch cells that require cell-based global net connections.

`-colFiller` Specifies that the cells you list with the `-cells` parameter are column filler cells.

`-isolation` Specifies that the cells you list with the `-cells` parameter are isolation cells.

`-minGaps` *T B L R*

Defines the distance, in microns, that must be reserved from the power domain boundary edges for power routing. You can specify four values, one for each edge.

`-name` *`newName`*

Specifies the new name for the power domain.

`pdName` Specifies the current name of the power domain to be modified. This is a positional parameter, and you must specify this value immediately after the command name.

`-powerSwitch` Specifies that the cells you list with the `-cells` parameter are isolation cells.

`-rowFiller` Specifies that the cells you list with the `-cells` parameter are isolations cells.

`-rowFlip {first | second | noFlip | auto}`

Flips the orientation of either the first or second row, then follows the flipped and abutted row pattern. The `noFlip` option preserves the R0 orientation for all rows.

The software automatically matches the orientation for rows inside the power domain with the orientation for rows outside the power domain. The auto option resets the software for this automatic (default) behavior if you have previously specified `first`, `second`, or `noFlip`.

Default: The software matches the row orientation inside and outside the power domain.

`-rowSpacing` *`value`* Specifies the row spacing between each row (1) or each pair of rows (2) as specified in `-rowSpaceType`.

`-rowSpacingType {1|2}`

Determines whether the row spacing value applies between each row (1) or each pair of rows (2).

`-rsExts` *`T B L R`*

Specifies the boundary for legal targets to be used by the power planning and routing commands, in conjunction with the power domain boundary. You can specify four values, one for each edge.

`-shifter` Specifies that the cells you list with the `-cells` parameter are shifter cells.

`-side {topBottom rightLeft}`

Specifies the well separator cells to place around the shifter or isolation cluster in a power domain if you have multiple-height cells or different separator cells. Instances of the cell listed first in the `-cells` list are placed at the top and bottom sides of the cluster. Instances of the cell listed second in the `-cells` list are placed at the right and left sides of the cluster. You cannot place different separators on all sides of the cluster.

You must specify `-wellSeparator` in addition to this parameter. You must specify two cells in the `-cells` list.

Default: If you do not specify this parameter, instances of the same cell are placed on all sides of the cluster.

`-splitter` Specifies that the cells you list with the `-cells` parameter are splitter cells.

`-wellSeparator` Specifies that the cells you list with the `-cells` parameter are well separator cells. You must use the `-side` parameter to specify which cells to use on the top and bottom, and right and left sides of the power domain.

Examples

- The following command changes the route search extension values of `PowerDomain1` to 15 μm on each side:

  ```
  modifyPowerDomainAttr PowerDomain1 -rsExts 15 15 15 15
  ```

- The following command places instances of well separator cells `well_single` on all sides of the shifter and isolation cell clusters in power domain `PD1`:

  ```
  modifyPowerDomainAttr PD1 -cells {well_single} -wellSeparator
  ```

- The following command places instances of well separator `well_single` on the top and bottom sides of the shifter and isolation cell clusters in power domain `PD2`, and instances of cell `well_double` on the right and left sides of `PD2`:

  ```
  modifyPowerDomainAttr PD2 -cells {well_single well_double} -wellSeparator
  -side {topBottom leftRight}
  ```

modifyPowerDomainMember

```
modifyPowerDomainMember pdName
     {-selected | -instances instanceList | -cells cellList}
     {-power {(VDD:VDD1 VDD2...)(VDD1:VDDL->PD1) (VDD2:VDDL->PD2)...}
     -ground {(GND:GND1 GND2...)...}}
     [-move]
     [-ioPins pin_list]
     [-cellForIOPin cellName]
```

Adds instances to an existing power domain, moves instances from one domain to another, and assigns I/O pads to a power domain.

Note: I/O pads assigned to a power domain are located outside the power domain boundary. However, they are highlighted whenever the associated power domain is highlighted.

Important

The `modifyPowerDomainMember` command is obsolete. To modify physical attributes of power domains, use the `modifyPowerDomainAttr` command.

Tasks such as power and ground specification are now done with CPF commands. To ensure compatibility with future releases, create a Common Power Format (CPF) file and include the related CPF commands.

For backward compatibility, the `modifyPowerDomainMember` command still works in this release. By default, the design created with this command cannot be saved to the Encounter database. To save an Encounter database, set the following variable:

```
setVar dbgSaveOldMsvDbOnly 1
```

The `modifyPowerDomainMember` command is order-dependent. If the specified instances of more than one `modifyPowerDomainMember` statement have a parent-child relationship, specify the most specific `modifyPowerDomainMember` statement before you specify the parent statement. For example:

```
modifyPowerDomainMember Domain1 -instances top_chip/module_A/instance_A -power
{(vdd1:VDD1)(vdd2:VDD2)} -ground {(vss1:VSS1)(vss2:VSS2)}
modifyPowerDomainMember Domain1 -instances top_chip/module_A -power (vdd1:VDD1)
-ground (vss1:VSS1)
```

If the power and ground connection of the specific `modifyPowerDomainMember` statement is exactly the same as the later one (its parent), the specific `modifyPowerDomainMember` statement can be skipped. For example, if there is no additional power and ground connection `(vdd2:VDD2) (vss2:VSS2)` for `top_chip/module_A/instance_A`, you can use the command as follows.

```
modifyPowerDomainMember Domain1 -instances top_chip/module_A -power (vdd1:VDD1)
-ground (vss1:VSS1)
```

Parameters

`-cellForIOPin` *`cellName`*

Specifies which I/O pin of a macro belongs to a specific power domain.

`-ground` {(*`GND:GND1 GND2`*...)...}

Specifies a list of ground nets and pin connections. If `-ground` has multiple ground nets, the software uses the last one in the list for tie-hi/lo connections. For example, if `-ground {(vss1:VSS1)(vss2:VSS2)}` is specified, then the software uses net `vss2` for tie-lo connection.

Note: You must specify either `-ground` or `-power` to ensure that the instance being moved contains the correct net:pin connections in the new power domain.

`-instances` *`instanceList`*

Specifies the names of the instances to be assigned to the power domain.

Note: If you want to add all unassigned instances to a power domain, you can specify `-instance *`. However, you must do this last, after assigning specific instances to power domains.

Note: For voltage level shifters that have not yet been inserted, you must specify the master name, not the instance name.

`-ioPins` *`pin_list`*

Assign I/O pins to power domains. This parameter mainly supports the bottom-up hierarchical design methodology. Because you have information about the power domains connected to the block I/Os, the tool automatically adds shifters to the nets entering or exiting the block, if the nets need level shifters. In addition to adding level shifters, this parameter also enables timing optimization to use correct buffering rules inside the block.

`-move` Moves the selected or named instances into the power domain.

pdName — Specifies the name of the power domain to which instances are to be added. You can use wildcards (* ?) with this parameter.

`-power` {(*VDD:VDD1 VDD2...*) (*VDD1:VDDL->PD1*) (*VDD2:VDDL->PD2*)...}

Specifies a list of power nets and pin connections.

For example, specify `-power (Vdd1:Vdd Pwr) (Vdd2:VDD vdd!)` to indicate that the `Vdd` and `Pwr` pins of all the instances in a given power domain are connected to the `Vdd1` net, and the `VDD` and `vdd!` pins are connected to the `Vdd2` net.

If `-power` has multiple power nets, the software uses the last one in the list for tie-hi/lo connections. For example, if `-power {(vdd1:VDD1)(vdd2:VDD2)}` is specified, then the software uses net `vdd2` for tie-hi connection.

The software enables the same type of level shifter in a power domain to connect to different power domains, for example:

```
modifyPowerDomain myPowerDomain instance inst1 -power
{(VDD:VDD) (VDD1:VDDL->PD1) (VDD2:VDDL->PD2)} -ground
(GND:GND)
```

Note: You must specify either `-ground` or `-power` to ensure that the instance being moved contains the correct net:pin connections in the new power domain.

`-selected` — Assigns the instances selected interactively in the design display area to the power domain.

Examples

- The following command moves the instances that you selected interactively into `PowerDomain1`:

  ```
  modifyPowerDomainMember PowerDomain1 -selected -move
  ```

- The following shifter table defines the relationship between the shifter cells and the power domains:

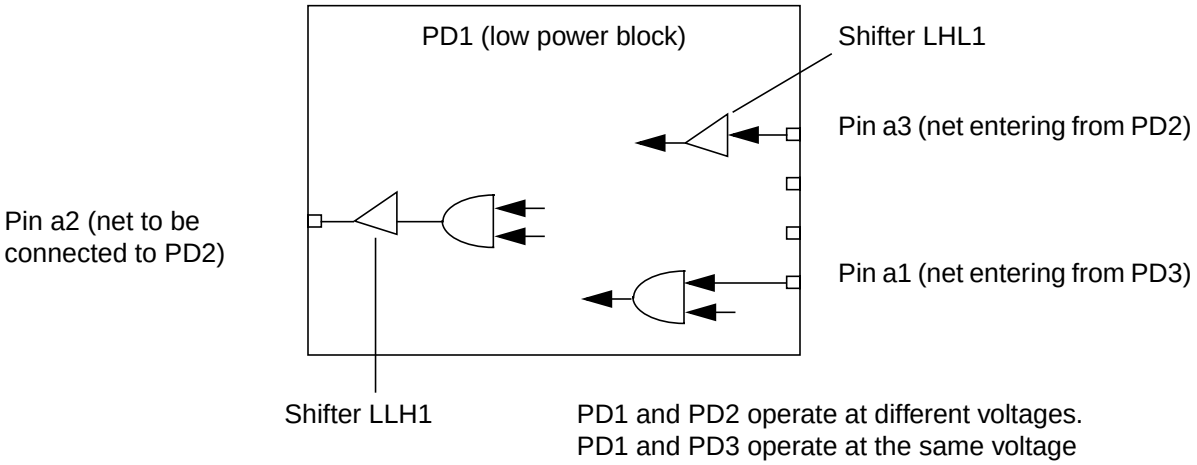
  ```
  LLH1    PD1    PD2    PD1
  LLL1    PD2    PD1    PD1
  ```

  The following command assigns I/O pin `a1` to power domain `PD3`:

  ```
  modifyPowerDomainMember PD3 -ioPins a1
  ```

The following command assign I/O pins a2 and a3 to power domain PD2:

```
modifyPowerDomainMember PD2 -ioPins a2 a3
```



- The following command assigns the A pin of macro MEM to power domain PD1:

```
modifyPowerDomainMember PD1 -ioPins A -cellForIOPin MEM
```

optPowerSwitch

```
optPowerSwitch
     [-commit {1|0}]
     [-effort {low | high}]
     [-maxIRDrop value_in_volts]
     [-maxSwitchIRDrop value_in_volts]
     [-net alwaysOnNetName]
     [-padFile fileName]
     [-readInstancePower fileName]
     [-readPowerSwitchCell fileName]
     [-reportFile fileName]
     [-setDontTouchCells cell_names]
     [-setDontTouchInstances instance_names]
     [-totalPower float]

     [-vsdgInFile fileName]
     [-fixViolations [0|1]]
     [-readPowerSwitchCell fileName]
     [-reportViolationsOnly fileName.rpt]
```

This command is used for two purposes: power switch optimization (reduction) and power switch ECO. Unless you specify `-vsdgInfile`, the command performs power switch optimization and reduces the number of switches needed in the rings and columns if possible.

For power switch optimization (reduction), this command starts with an existing power switch configuration, then removes switches to create an optimum power switch configuration that meets the criteria you set. The tool then places the switches.

In this case, the following parameter is required:

- `-readPowerSwitchCell`

You can specify the following optional parameters for power switch optimization:

- `-commit`
- `-effort`
- `-maxIRDrop`
- `-maxSwitchIRDrop`
- `-net`
- `-padFile`
- `-readInstancePower`
- `-reportFile`

- `-totalPower`

If you specify `-vsdgInFile` with any of these commands, the command stops and issues an error message.

If you specify `-vsdgInFile`, the tool does power switch ECO. The `optPowerSwitch` command reads a binary-format report generated by VSDG to identify Id_sat violations caused by current exceeding the threshold value for each switch, then repairs the violations. This command can fix violations in the following ways:

- Replacing a filler with a switch
- Replacing a switch with a bigger switch

Note: In this case, the inserted power switches could overlap standard cells. After running `optPowerSwitch`, run `refinePlace` to move the standard cells away from the switches.

If you specify `-vsdgInFile`, you can only use the following parameters:

- `-reportFile`
- `-fixViolations`
- `-reportViolationsOnly`

Parameters

`-commit {1|0}`

Applies the power optimization results to the design. By default, the tool uses the results of power switch optimization to place the switches.

Default: `1`

`-effort {low|high}`

Determines the power switch optimization iterations. If you specify `-low`, the tool stops the optimization process after finding in initial optimization. By default, the tool continues to iterate optimization until the best results are found.

Default: `high`

`-fixViolations [0|1]`

Repairs placement and routing problems found during power switch optimization by refining the shifter and isolation cell placement.

Default: `1`

Note: You can use this parameter for power switch ECO only. You must also specify `-vsdgInFile`.

`-maxIRDrop` *`value_in_volts`*

Specifies the maximum IR drop allowed inside power domains.

`-maxSwitchIrDrop` *`value_in_volts`*

Specifies the maximum IR drop allowed for switches.

`-net` *`alwaysOnNetName`*

Specifies the always-on net to use if there is more than one always-on net. If you do not use this parameter, the tool searches for the always-on net.

`-padFile` *`fileName`*

Specifies the name of the file containing a list of pads.

`-readInstancePower` *`fileName`*

Specifies the name of the containing a list of power switch instances for power switch prototyping.

`-readPowerSwitchCell` *`fileName`*

Specifies the name of the file containing the power switch cell parameters. This is the same file that Encounter® Power System (Next-Generation VoltageStorm) uses for Pi analysis.

`-reportFile` *`fileName`*

For power switch optimization:

Creates a file containing Encounter commands for removing the identified switches.

For power switch ECO:

If you specify `-vsdgInFile` together with this parameter, the output file that contains the changes made – replacing a filler with a switch or upsizing a switch. This file contains error and warning messages related to the changes that were not possible to make without violating a current constraint, and lists the constraint that would be violated if the change were made. The file also contains the number of Id_sat violations identified for a switch instance, and the corrective action taken.

`-reportViolationsOnly` *`fileName`*`.rpt`

Reports violating instances to file *`fileName`*`.rpt` and highlights them in the artwork window.

Note: You can use this parameter for power switch ECO only. You must also specify `-vsdgInFile`.

`-setDontTouchCells {`*`cell_names`*`}`

Prevents the tool from removing instances of the specified cells when optimizing the number of power switches. This parameter enables you to preserve cells that are used for other purposes, such as level shifting, in addition to power switching.

`-setDontTouchInstances {`*`instance_names`*`}`

Prevents the tool from removing the specified instances when optimizing the number of power switches. This parameter enables you to preserve cells that are used for other purposes, such as level shifting, in addition to power switching.

`-totalPower` *`float`*

Specifies a target value for the power consumed by the power switch ring.

`-vsdgInFile` *`fileName`*

Specifies the VSDG binary-format report to read. The command uses this file to find power switch cells that violate physical design constraints.

Note: You can use this parameter for power switch ECO only.

Examples

To obtain the report you need as input to `optPowerSwitch`, complete the following steps:

1. Precharacterize power gating cells in `Libgen`
2. Run VoltageStorm:

   ```
   runVStorm \
   -dynamic \ #dynamic current
   -analyzePI 1 \ #power switch analysis
   ... \ #other VoltageStorm commands you want to run
   ```

   The `runVStorm` command writes power switch analysis results in a file named `vs2fe_pg.rpt`.

Sample VoltageStorm ECO File (vs2fe_pg.rpt)

```
VERSION ?1.0?;
PGCELL 3;
-HEAD16DM 1.044590e-02;
-HEAD32DM 2.044590e-02;
-HEAD8DM 5445895e-02;
END;
PGINST 378;
-DTMF_INST/TDSP_CORE_INST/pso1_TDSPCore_1_HEAD16DM_560_276_0 HEAD16DM
6.574561e-01 1.585955e-02 ;
```

Sample optPowerSwitch Output File

```
Violating instance: DTMF_INST/TDSP_CORE_INST/pso1_TDSPCore_1_HEAD16DM_560_276_0
(HEAD16DM)
Actual current: 0.01585955A
Current capacity: 0.01044591
Fixes: Replace DTMF_INST/TDSP_CORE_INST/pso1_TDSPCore_1_HEAD16DM_560_276_0
(HEAD16DM) ->
DTMF_INST/TDSP_CORE_inst/pso1_TDSPCore_1_HEAD32DM_559_276_0 (EHAD32DM)
Current capacity: 0.0104459A -> 0.02.4459A
```

```
Violating instance: -DTMF_INST/TDSP_CORE_INST/
pso1_TDSPCore_1_HEAD16DM_560_635_125 (HEAD16DM)
Actual current: 0.01189175A
Current capacity: 0.0104459A
Fixes: Replace DTMF_INST/TDSP_CORE_inst/pso1_TDSPCore_1_HEAd16DM_559_276_0
(HEAD16DM) -> DTMF_INST/TDSP_CORE_inst/pso1_TDSPCore_1_HEAd32DM_559_276_0
(HEAD32DM)
Current capacity: 0.0104459A -> 0.0204459A
```

Sample Violation Report

```
optPowerSwitch -vsdgInFile <your file/path vs2fe_pg.rpt> -reportViolationsOnly
PSOcurrent.rpt
```

The `PSOcurrent.rpt` file contains the following information:

```
PSO Instance name : Isat(actual) : Isat(capacity) : Diff. in current : Status Pass/
Fail

DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_560_276_0 : 0.01585955
(0.0104459) : -0.00541365 : FAIL
DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_560_635_125 : 0.01189175
(0.0104459) : -0.00144585 : FAIL
DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_710_276_126 : 0.007183449
(0.0104459) : 0.003262451 : PASS
DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_560_462_65 : 0.006478845
(0.0104459) : 0.003967055 : PASS
DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_560_465_66 : 0.006441807
(0.0104459) : 0.004004093 : PASS
DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_560_454_62 : 0.006307256
(0.0104459) : 0.004138644 : PASS
DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_560_279_1 : 0.006262664
(0.0104459) : 0.004183236 : PASS
DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_560_282_2 : 0.006247832
(0.0104459) : 0.004198068 : PASS
DTMF_INST/TDSP_CORE_INST/psoI_TDSPCore_1_HEAD16DM_560_451_61 : 0.006232314
(0.0104459) : 0.004213586 : PASS
```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"
  - "Power Switch Optimization"

reorganizeFanout

```
reorganizeFanout
     -nets {list_of_nets}
```

Balances the number of fanouts the specified nets drive and reorders the fanouts each net drives based on the fanouts' physical location. After reordering, each of the specified nets drives roughly an equal number of fanouts. The approach reduces routing congestion and reduces peak power consumption when a block is powered on from the shut-off state.

Use this command after placement.

Parameters

`-nets {list_of_nets}`

Specifies the nets whose fanouts the command must balance and reorganize.

Examples

The following command balances and reorganizes the fanouts of `net1`, `net2`, and `net3`:

```
reorganizeFanout -nets {net1 net2 net3}
```

repairPowerDomain

```
repairPowerDomain
     -CpfRules ruleNames
```

Repairs power domains by applying CPF rules after operations such as timing optimization, which can add buffers.

Parameters

`-CpfRules`	Re-applies the CPF rules that match the specified rule names.

Examples

The following command applies CPF rules that begin with `iso`:

```
repairPowerDomain -CpfRules iso*
```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"
  - "Support for the Common Power Format"

replaceAssignBuffer

```
replaceAssignBuffer
     -powerDomain powerDomain
     -newBuffer bufferName
```

Replaces assign buffers inside a power domain with a different buffer.

Parameters

`-newBuffer` *`bufferName`*

Specifies the replacement buffer.

`-powerDomain` *`powerDomain`*

Specifies the power domain in which you want to replace the assign buffers.

Examples

The following command replaces assign buffers in `PD1` with buffer `BUF1`:

```
replaceAssignBuffer -powerDomain PD1 -newBuffer BUF1
```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"
  - "Data Preparation"

replacePowerSwitch

```
replacePowerSwitch
     -insts inst_name_list
     -cell cell_name
     [-xoffset value_in_microns]
     [-yoffset value_in_microns]
     [-orientation orienation_syntax]
```

Replaces the cell of a power switch instance. You can also change the orientation of the instance after cell substitution, and provide a new location for the instance origin relative to the original instance origin.

Parameters

`-cell` *`cell_name`*

Specifies the cell to substitute for the instance's cell.

`-insts` *`inst_name_list`*

Specifies the instance whose cell you want to replace.

`-orientation` *`orientation_syntax`*

Specifies the orientation of the instance after the cell substitution. Choose one of the following values: `R0`|`R90`|`R180`|`R270`|`MX`|`MY`|`MX90`|`MY90`

Default: The same orientation as the original instance.

`-xoffset` *`value_in_microns`*

Specifies the instance placement in the x direction after the cell substitution. A positive value moves the instance to the right of the original instance origin. A negative value moves the instance to the left of the original instance origin.

`-yoffset` *`value_in_microns`*

Specifies the instance placement in the y direction after the cell substitution. A positive value moves the instance above the original instance origin. A negative value moves the instance above the original instance origin.

Examples

- The following example substitutes cell `CDN_RING_SW1` for the original cell for instance `pso1_TDSP2_1_CDN_RING_SW_636_1309_6`:

  ```
  replacePowerSwtich -insts pso1_TDSP2_1_CDN_RING_SW_636_1309_6 -cell
  CDN_RING_SW1
  ```

- The following example substitutes cell `CDN_RING_SW1` for the original cell for instance `pso1_TDSP2_1_CDN_RING_SW_636_1315_8` and changes the orientation of the instance to `MY90`:

  ```
  replacePowerSwitch -insts pso1_TDSP2_1_CDN_RING_SW_636_1315_8 -cell
  CDN_RING_SW1 -orientation MY90
  ```

reportIsolation

```
reportIsolation
     [-highlight]
     [-outfile fileName]
     [-fromPowerDomain powerDomain]
     [-toPowerDomain powerDomain]
```

Reports or shows the added isolation cells and instances.

Note: Use this command after you add isolation cells. For more information, see addIsolationCell on page 271.

Parameters

`-fromPowerDomain`	Reports isolation cells in the driving power domain. You must use this parameter with `-toPowerDomain`. *Default:* Reports all isolation cells.
`-highlight`	Highlights isolation cells and instances in the main window.
`-outfile` *`fileName`*	Writes isolation cell and instance names to a report. *Default:* The software writes the report to a default file: *`designName`*`.isolation.rpt`
`-toPowerDomain`	Reports isolation cells in the receiving power domain. You must use this parameter with `-fromPowerDomain`. *Default:* Reports all isolation cells.

Examples

The following command highlights isolation cells and writes the cell names to `isolationCellReport`:

```
reportIsolationCell -highlight -outfile isolationCellReport
```

reportPowerDomain

```
reportPowerDomain
     -powerDomain powerDomainName
     -file outFile
     -module {list_of_modules}
     -shifter
     -isoInst
     -pgNet
     -bindLib
     -inst inst_name
```

Reports information about power domains. The report includes the following attributes:

- Power domain name
- Power domain bounding box
- Timing library
- Mingap
- RS extension
- Row type
- Row spacing

The command also reports whether the power domain is always on.

Parameters

`-bindLib`	Reports the timing library bound to each instance of a power domain.
`-file` *`outFile`*	Specifies the name of the output file. *Default:* *`designName`*`-powerdomain.rpt`
`-inst` *`inst_name`*	Reports the power domain-related information for the specified instance: the input, output, and power/ground connections.
`-isoInst`	Reports all isolation cells in the power domain.
`-module {list of modules}`	Checks whether the specified instances or hierarchical instances are in the power domain. *Default:* The software does not perform this check.

`-powerDomain` *`powerDomainName`*

Specifies the power domain whose attributes you want to report.

Default: If you do not specify the power domain, this command report data about all power domains.

`-pgNet` Reports all the power/ground nets in the power domain.

`-shifter` Reports all shifters in the power domain.

Examples

- The following command reports all shifter cells, all isolation cells, and all power/ground nets in power domain `PD1`. The command also checks whether HInst `instA` is contained in `PD1`, and reports all results to `myDesign-powerdomain.rpt`.

```
reportPowerDomain pdName PD1 -file myDesign-powerdomain.rpt \
    -module {instA} -shifter -isoInst -pgNet
```

- The following example shows how output file `outfile` contains the binding between the timing libraries and instances in power domain VCORE:

```
-reportPowerDomain -bindLib -powerDomain VCORE -file outfile


Timing library for instance(s):
==========================
INST = uLSCVSocDownVCore/uACLKEND/uLVHL (lvl_slow_1v08_1v08)
INST = uLSCVSocDownVCore/uACLKENI/uLVLHL (lvl_slow_1v08_1v08)
INST = uLSCVSocDownvCore/uaCLKENP/uLVLHL (lvl_slow_1v08_1v08)
INST = uLSCVSocDownvCore/uaCLKENRW/uLVLHL (lvl_slow_1v08_1v08)
INST = uLSCVSocDownvCore/uaBIGENDINIT/uLVHlL (lvl_slow_1v08_1v08)
INST = uLSCVSocDownvCore/uaCLKIN/uLVHL (lvl_slow_1v08_1v08)
INST = uLSCVSocDownvCore/uCP15SDISABLE/uLVHL (lvl_slow_1v08_1v08)
```

- The following command reports the power domain-related information about instance `PM_INST/state_inst/g45`:

```
reportPowerDomain -inst PM_INST/state_inst/g45
```

Result:

```
Analyzing PM_INST/state_inst/g45 (Cell: AN2D1) power domain:AO
------from side ------
Pin A2 => Net PM_INST/state_inst/n_28
PM_INST/state_inst/g71/ZN in power domain A0
Pin A1 => Net PM_INST/state_inst/n_25
PM_INST/state_inst/g47/Z in power domain A0
------to side ------
Pin Z => Net PM_INST/state_inst/n_36
```

```
PM_INST/state_inst/fsm_reg[1]/qi_reg/D in power domain A0
==========PG pins =============
VDD conneted to Net: VDD
VSS connected to Net:VSS
```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"

reportPowerSwitch

```
reportPowerSwitch
     -outFile file_name
```

Reports each power switch instance, the location, and pin/net connections. The report format is as follows:

```
#InstName (x,y)
#  (pin1): net1
#  (pin2): net2
```

Parameters

`-outFile` *`file_name`*

Specifies the name of the report file where the information is written.

reportShifter

```
reportShifter
     [-cell cellName]
     [-fromPD powerDomainName]
     [-toPD powerDomainName]
     [-outfile fileName]
     [-highlight]
```

Reports the level shifters that have been read into or defined for the Encounter software.

Use this command after you use the loadShifter and addShifter commands. If you want to highlight level shifters, you must first run placement.

Parameters

`-cell`
: Reports shifters with the specified cell.

  Default: Reports all level shifters.

`-fromPD` *`powerDomainName`*
: Reports shifters in the driving power domain.

  Default: Reports all level shifters.

`-highlight`
: Highlights placed level shifters.

`-outfile` *`fileName`*
: Reports all instances in an output file.

  Default: The instances will be reported to a default output file *`designName`*`.shifter.rpt`. A warning message is printed to indicate a default output file is created.

`-toPD` *`powerDomainName`*
: Reports shifters in the receiving power domain.

  Default: Reports all level shifters.

Examples

The following command reports all level shifters in design `chip`:

```
loadShifter -infile chip.vsf
addShifter
reportShifter
```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"
  - "Place Standard Cells and Macros"

routePGPinUseSignalRoute

```
routePGPinUseSignalRoute
     [-nets {listofNets}
     [-pattern {trunk | steiner}]
     [-maxFanout number]
     [-nonDefaultRule ruleName]
```

Routes the secondary power/ground pins of always-on cells to the specified power/ground nets in signal route. You can also control the routing pattern and wire by specifying the pattern, the maximum number of pins per cluster, and non-default rules.

Parameters

`-nets listOfNets`
: Specifies the power/ground nets to be connected to the secondary power/ground pins. Reports shifters with the specified cell.

  Default: If this parameter is not specified, the software will get all the nets from global connections for the cells and pins defined in the command.

`-pattern {trunk | steiner}`
: Specifies how the signal router routes the secondary power/ground pins.

  Trunk: When value is 1, routes the pins to nearest power grid. When value is more than 1, routes all pins per cluster to the trunk.

  Steiner: Routes all pins to a trunk.

  Default: `trunk`

`-maxFanout number`
: Specifies the maximum number of pins per cluster in the trunk rotating pattern.

  Default: `10`.

`-nonDefaultRule ruleName`
: Specifies the non-default rule the signal router uses to route the secondary power/ground pins.

  Default: Regular signal routing rule is used by default.

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"

saveCPF

```
saveCPF
     fileName
     [-hierInst hinstName]
     [-inst instName]
     [-topHier]
     [-update]
```

Writes native Encounter power-related information to the specified file.

Parameters

fileName	Specifies the name of the CPF output file. If you do not specify `-update`, then the software writes the original CPF file.
`-hierInst` *hinstName*	Creates a CPF file for the specified hierarchical instance.
`-inst` *instName*	Creates a CPF file for the specified block.
`-topHier`	Creates a CPF file for the top level design.
`-update`	Updates the CPF file to include any implementation changes you have made to the design during an interactive Encounter session.

Example

- The following command saves the `low_power.cpf` file:

  ```
  saveCPF low_power.cpf
  ```

- The following command saves a CPF file for block B:

  ```
  saveCPF myCPFFile -inst B
  ```

- The following command saves a CPF file for the top level design:

  ```
  saveCPF myTopCPFFile -topHier
  ```

- The following command saves a CPF file for hierarchical instances A, B, and C:

  ```
  saveCPF myHierCPFFile -hierInst {A B C}
  ```

- The following command updates file `low_power.cpf:`

  ```
  saveCPF -update low_power.cpf
  ```

Related Topics

- CPF 1.0 Script Example" chapter in the *Encounter User Guide*
- CPF 1.0e Script Example" chapter in the *Encounter User Guide*
- Supported CPF 1.0 Commands" chapter in the *Encounter User Guide*
- Supported CPF 1.0e Commands" chapter in the *Encounter User Guide*

setPGPinUseSignalRoute

```
setPGPinUseSignalRoute
     [-off]
     cellName:pinName cellName:pinName2 ...
```

Sets the internal bit for a given cell/pin pair to be routed by the signal router. Use this command to enable NanoRoute to route always-on-pins in SRPG (state retention power gating) or always-on buffer cells.

Parameters

cellName	Specifies the cell containing the pin for NanoRoute to route. You can use a wildcard to specify all cells of certain type.
-off	Unsets this feature.
pinName	Specifies the pin to be routed.

Examples

- The following example sets both the vddo and vsso pins for the signal router, but NanoRoute will only route vddo:

  ```
  setPGPinUseSignalRoute SDRFF:vddo SDRFF:vsso
  routePGPinUseSignalRoute -nets VDD
  ```

- The following example sets NanoRoute to route the VDD1 pin of all NAND cells:

  ```
  setPGPinUseSignalRoute NAND*:VDD1
  ```

specifyIsolationCell

```
specifyIsolationCell
     -cell leafCellName |
     -group groupName |
     -hinst hinstName |
     -inst instName |
     [-prefix prefix]
```

Specifies the instances to use as isolation cells.

Note: You must first import the design.

Parameters

`-cell` *`leafCellName`*

Designates all instances of the specified cell as isolation cells.

`-group` *`groupName`*

Designates all members on the specified group as isolation cells.

`-hinst` *`hinstName`*

Designates all instances of the specified hierarchical instance as isolation cells.

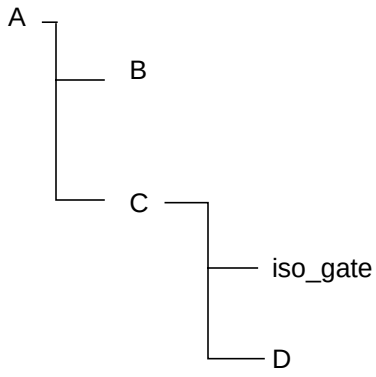
`-inst` *`instName`*

Designates the specified instance as an isolation cell.

`-prefix` Designates only the instances that match the prefix as isolation cells. This allows you to narrow the selection of cells within specified by `-cell`, `-hinst`, or `-group`.

Examples

The following example shows a module `A`, containing modules `B` and `C`. Module `C` contains `iso_gate` and `D`.



- The following command specifies all instances in the `A/C` module as isolation cells:

  ```
  specifyIsolationCell -hinst A/C
  ```

  `iso_gate` is specified as an isolation cell.

- The following command specifies instance `A/C/iso_gate` as an isolation cell.

  ```
  specifyIsolationCell -inst A/C/iso_gate
  ```

  `iso_gate` is specified as an isolation cell.

- The following command specifies all instances in module `A/C` whose prefix is `iso_` as isolation cells:

  ```
  specifyIsolationCell -hinst A/C -prefix iso_
  ```

  `iso_gate` is specified as an isolation cell.

unspecifyIsolationCell

```
unspecifyIsolationCell
     -cell leafCellName |
     -group groupName |
     -hinst hinstName |
     -inst instName |
     [-prefix prefix]
```

Undoes the effect of `specifyIsolationCell`. For more information, see specifyIsolationCell on page 402.

Parameters

`-cell` *`leafCellName`*

Instances of the specified cell are no longer designated as isolation cells.

`-group` *`groupName`*

Members of the specified group are no longer designated as isolation cells.

`-hinst` *`hinstName`*

Instances of the specified hierarchical instance are no longer designated as isolation cells.

`-inst` *`instName`*

The specified instance is no longer designated as an isolation cell.

`-prefix` Instances of specified cells, HInsts, and instances, and members of the specified group are no longer designated as isolation cells.

Examples

- The following command instructs the software to stop designating all instances in the `A/C` module as isolation cells:

  ```
  unspecifyIsolationCell -hinst A/C
  ```

- The following command instructs the software to stop designating instance `A/C/iso_gate` as an isolation cell.

  ```
  unspecifyIsolationCell -inst A/C/iso_gate
  ```

- The following command instructs the software to stop designating all instances in module `A/C` whose prefix is `iso_` as isolation cells:

```
unspecifyIsolationCell -hinst A/C -prefix iso_
```

verifyPowerDomain

```
verifyPowerDomain
     [-allInstInPD]
     [-bind]
     [-gconn [module_name_list]]
     [-isoNetPD [fileName]
     [-noFTermIsoC]]
     [-noFTermIsoC]
     [-noFTermShifter]
     [-place [-place_rpt report_name]]
     [-powerSwitch]
     [-pwrwithinmingap]
     [-xNetPD [fileName] [-noFTermShifter]]
```

Verifies whether the power domains are set up correctly. This command always checks whether any power domains overlap. The `verifyPowerDomain` command checks that libraries specified for each power domain of each `dc_corner` are complete for each power domain member list.

Parameters

`-allInstInPD`	Verifies all instances are in the power domain.
`-bind`	Verifies that all cells are bound to correct timing library cells.
`-gconn` *`module_name_list`*	Verifies that power and ground pins of instances within the power domain are connected to the power and ground nets of that power domain. In addition, this parameter verifies that the tie-high and tie-low connections of a cell are connected to the same power and ground nets of the power domain to which the cell belongs. If you do not specify a list of module names, all modules are verified.
`-isoNetPD`	Verifies that nets crossing power domains have isolation cells, and checks whether the isolation cell placement matches the specification in the shifter table. Before using `verifyPowerDomain -isoNetPD`, read the shifter table into the software using the `loadShifter` command.
`-noFTermIsoC`	Ignores I/O nets when verifying isolation cells.
`-noFTermShifter`	Ignores I/O pin nets when verifying shifter cells.
`-place`	Verifies that only those cells assigned to a power domain are placed within the boundary of that power domain.

`-place_rpt` *`report_name`*

Specifies a name and location for the report generated when you use this command with the `-place` parameter.

`-powerSwitch` Verifies whether each row in the power domains contain at least one power switch. The command issues a warning message detailing which rows do not contain at least one switch.

`-pwrwithinmingap` Verifies that the power domain ring is within the distance specified by the `modifyPowerDomainAttr -minGaps` or `createPowerDomain -minGaps` command.

`-xNetPD` Reports which nets that cross power domains require a level shifter to be inserted, and checks whether the shifter cell placement matches the specification in the shifter table. Before using `verifyPowerDomain -xNetPD`, read the shifter table into the software using the `loadShifter` command.

Examples

The following command issues warning messages if cells that are not members of a power domain are placed within that power domain boundary, or if any power domains overlap:

```
verifyPowerDomain -place
```

Related Topics

- Low Power Design chapter of the *Encounter User Guide*
  - "Overview"
  - "Power Domain Shutdown and Scaling"
  - "Multiple Supply Voltage Flat Flow"
  - "Verify Power Domains"

verifyPowerSwitch

```
verifyPowerSwitch
     [-powerDomain powerDomain]
```

Analyzes whether switchable power domains contain at least one switch per row. The command issues a warning message detailing which rows do not contain power switches. The `verifyPowerDomain -powerSwitch` parameter performs the same function.

Parameters

`-powerDomain` *`powerDomain`*

Verify only the specified power domain.

Examples

The following command verifies that each row in power domain PD1 contains at least one switch:

```
verifyPowerSwitch -powerDomain PD1
```

15

Metal and Via Fill Commands

addMetalFill

```
addMetalFill
     [-area x1 y1 x2 y2]
     [-excludeVias {[{list_of_via_names}] [{list_of_via_rules}]}]
     [-iterationNameList {list_of_iteration_names}]
     [-keepSquareFloating]
     [-layer {layerNumber | {list_of_layer_numbers}]
     [-onCells [-std]]
     [-removeFloatingFill]
     [-slackThreshold slack]
     [-snap]
     [-squareShape] | -nets {netNameList} [-genViaOnly] [-mesh] ]
     [-timingAware {on | off | sta}]
     [-stagger {on | off | diag}]
```

Inserts inactive metal into a placed and routed design to achieve the metal density within the range required by a specific manufacturing process. Avoids inserting metal on top of macros if it can achieve the preferred metal density without doing so.

During timing-driven metal fill, the software assigns costs to signal and clock nets so it can avoid placing metal fill near timing-critical nets.

- Add metal fill after adding via fill.
- If you use the `-timingAware sta` parameter, you must run `report_timing` before adding metal fill.

This command uses the values specified by the `setMetalFill` command. For information, see setMetalFill on page 427.

Adding Metal Fill in Multi-Threading Mode

You can add metal fill in multi-threading mode by using the following command before adding the metal fill:

- `setMultiCpuUsage`

For more information, see the following documents:

- Multiple-CPU Processing Commands chapter of the *Encounter User Guide*.
- “Accelerating the Design Process by Using Multiple-CPU Processing” chapter of the *Encounter User Guide*

Parameters

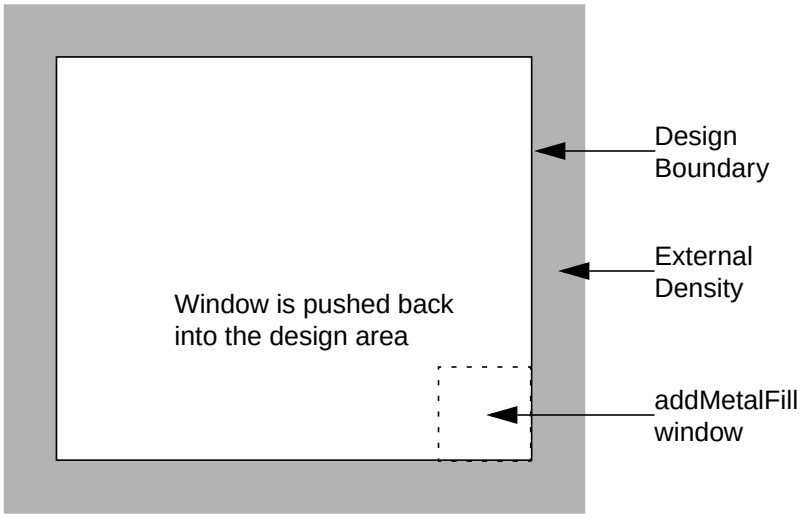
`-area` *x1 y1 x2 y2*

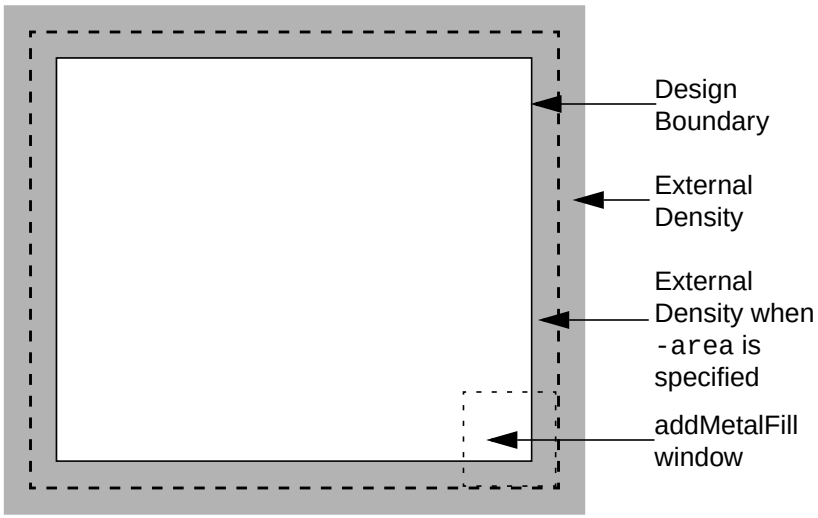
Specifies a set of coordinates within which to add metal fill. This command reduces the fill area by half of the maximum design rule spacing defined in the LEF file. For example, if you specify `-area 0 0 100 100`, and the maximum spacing rule in the LEF file is 2.5 µ, `addMetalFill` adds fill in the following area: `1.25 1.25 98.75 98.75`.
Default: If you do not specify this parameter, adds metal fill to the entire design area (taking the maximum spacing rule into account).

Behavior of addMetalFill -area with external metal density

When metal fill is added to a block design the external metal density provides a density assumption to `addMetalFill`. The external density is used only for density calculation purposes and no fill shapes are added outside of the design area.

When the `-area` parameter is not specified, the external density value is not used. In this case, the `addMetalFill` window is pushed back if it extends outside the design area, as shown in the figure below.





If the `-area` parameter is specified and is larger than the design size then `addMetalFill` windows extend outside the block area and use the external density value (see setMetalFill on page 427 for external density value specification) for that area.

If external density is less than the specified preferred density, it is more likely that shapes will be added to the area. If external density is greater than the preferred density it is less likely that shapes will be added (shapes are only added within the design boundary).

Note: Windows outside the design boundary are ignored.

`-excludeVias {[{`*`list_of_via_names`*`}] [{`*`list_of_via_rules`*`}]}`

Specifies vias or via rules to exclude from use by metal fill. Use this parameter to prevent the software from using the same vias inside MACROs and for metal fill insertion.

Separate the arguments with spaces. You can specify both via names and via rules.

- The following example shows how to specify a list of via names:

  ```
  -excludeVias {VIA12 VIA23}
  ```

- The following example shows how to specify a list of via rules:

  ```
  -excludeVias {via1Array via2Array}
  ```

- The following example shows how to specify both via names and via rules:

  ```
  -excludeVias { VIA12 VIA23 via1Array via2Array}
  ```

`-genViaOnly`

Specifies that only generated vias are used for metal fill connections.

Note: This parameter is enabled only when the `-nets` parameter is specified.

`-iterationNameList {`*`list_of_iteration_names`*`}`

Specifies a list of iterations (the names are specified by the `setMetalFill -iterationName` parameter) for adding metal fill. Processes the iterations in the order listed, and stops when the preferred density is reached in any iteration.

Running `addMetalFill` without the `-iterationNameList` parameter is the same as running the following command:

```
addMetalFill -iterationNameList default
```

If `addMetalFill` is looking for `default`, that is, if `-iterationNameList default` is specified or if you do not specify the `-iterationNameList` parameter, and the default settings are not defined by `setMetalFill`, the `addMetalFill` command uses it own internal default values, which are not derived from any `setMetalFill -iterationName` values.

When you run `addMetalFill` in iteration mode, it runs several times with just one invocation of the command, and uses the ending density from an iteration as the starting density of the next iteration. For that one invocation, the window size and step for a specific layer must be the same for all iterations.

In the following example, `addMetalFill` fails because the specified values for `-windowSize` and `-windowStep` in `step1`, `step2`, and `step3` are different:

```
setMetalFill -iterationName step1 -layer ... \
  -windowSize 100 100 -windowStep 50 50
setMetalFill -iterationName step2 -layer ... \
  -windowSize 100 100 -windowStep 50 50
setMetalFill -iterationName step3 -layer ... \
  -windowSize 50 50 -windowStep 25 25

addMetalFill -iterationNameList {step1 step2 step3} ...
```

If you run `addMetalFill` two times, and separate `step3`, (because its values are different from those of `step1` and `step2`) the commands will succeed:

```
addMetalFill -iterationNameList {step1 step2} ...
addMetalFill -iterationNameList step3 ...
```

`-keepSquareFloating`

Specifies that unconnected (floating) metal fill must be square.
Default: If you do not specify this parameter, the software can create rectangular and square metal fill that is unconnected.

`-layer` *layerNumber* | **{***list_of_layer_numbers***}**

Specifies the layers to which to add metal fill. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks. To specify metal layers 1, 2, and 3, type one of the following commands:

```
addMetalFill -layer "1 2 3"
addMetalFill -layer {1 2 3}
```

Default: If you do not specify this parameter, adds metal fill to all layers.

`-mesh`

Connects metal fill shapes to the power mesh, allowing it to carry current as part of the power and ground structure. With mesh connections the software uses the maximum number of cuts in vias, based on the intersection area between layers, instead of the following the minimum-cut rule used with tree connections. Increasing the number of cuts helps to reduce IR drop.

> **Note:** This parameter is enabled only when the `-nets` parameter is specified.

The default values for the metal fill commands are optimized for timing, not power. For recommendations for values that are optimized for power, see "Recommendations for Power Strapping Mode" in the "Optimizing Metal Density" chapter of the *Encounter User Guide*.

`-nets` **{***netNameList***}**

Specifies a list of special nets to which to connect metal fill. The nets must be listed in the `SPECIALNETS` section of the DEF file. The software attempts to connect to the first net in the list, then the next net, and so forth.
Default: If you do not specify this parameter, metal fill is not connected to special nets (it is floating).

Note: Do not specify this parameter if you also specify `-squareShape`.

`-onCells [-std]]`

Allows the software to add metal fill to `CLASS BLOCK` and `CLASS PAD` macro cells.

Before adding the metal fill, make sure that the cells are at the preferred density, because adding metal fill shapes inside the cells might lead to timing problems inside the cells.

Note: The software can always add metal fill to `CLASS CORE SPACER` cells, but can add it to `CLASS BLOCK` and `CLASS PAD` cells only when this parameter is specified.

`-std`

Adds metal fill to standard cells only.

`-removeFloatingFill`

Removes unconnected metal fill from the design. When the software adds metal fill, it creates both connected and unconnected fill. When you specify `-removeFloatingFill`, it removes all the unconnected fill and leaves only metal fill that is connected.
Default: If you do not specify this parameter, the software leaves floating metal fill in the design.

`-slackThreshold` *`slack`*

Specifies a slack threshold, in nanoseconds, for critical nets during timing-driven metal fill when you specify `-timingAware sta`. All nets with slack less than the `-slackThreshold` are considered critical.

For most designs, Cadence recommends you specify a small positive value for the slack threshold, such as 100 picoseconds, which gives you a small safety margin.
Default: `0.0`

`-snap`

Snaps edges of metal fill shapes and centers of cuts to a user-defined snap grid that is a multiple of the manufacturing grid. When the shapes are snapped to this grid, the edges of the shapes lie on the manufacturing grid.

Other Encounter features share the grid, including verify, power planning and routing, and wire editing. Define the grid using the `setSnapGrid` command. If `setSnapGrid` is not specified, the software snaps the shapes to the manufacturing grid.

For more information, see setSnapGrid on page 2261.

Note: When you specify this parameter, the software assumes that vias from the LEF file meet the snap grid requirements.

`-squareShape`

Specifies that only square-shaped metal fill is added.
Default: If you do not specify this parameter, both rectangular and square metal fill can be added.

Note: Do not specify this parameter if you use the `-nets` parameter. If you specify both parameters, the software ignores `-nets` and inserts floating square metal fill.

`-stagger {on | off | diag}`

Specifies the pattern for adding metal fill.
Default: `on`

Specify one of the following parameters:

`on`

Adds metal fill shapes in a staggered pattern in the preferred routing direction and in an aligned pattern in the non-preferred direction.

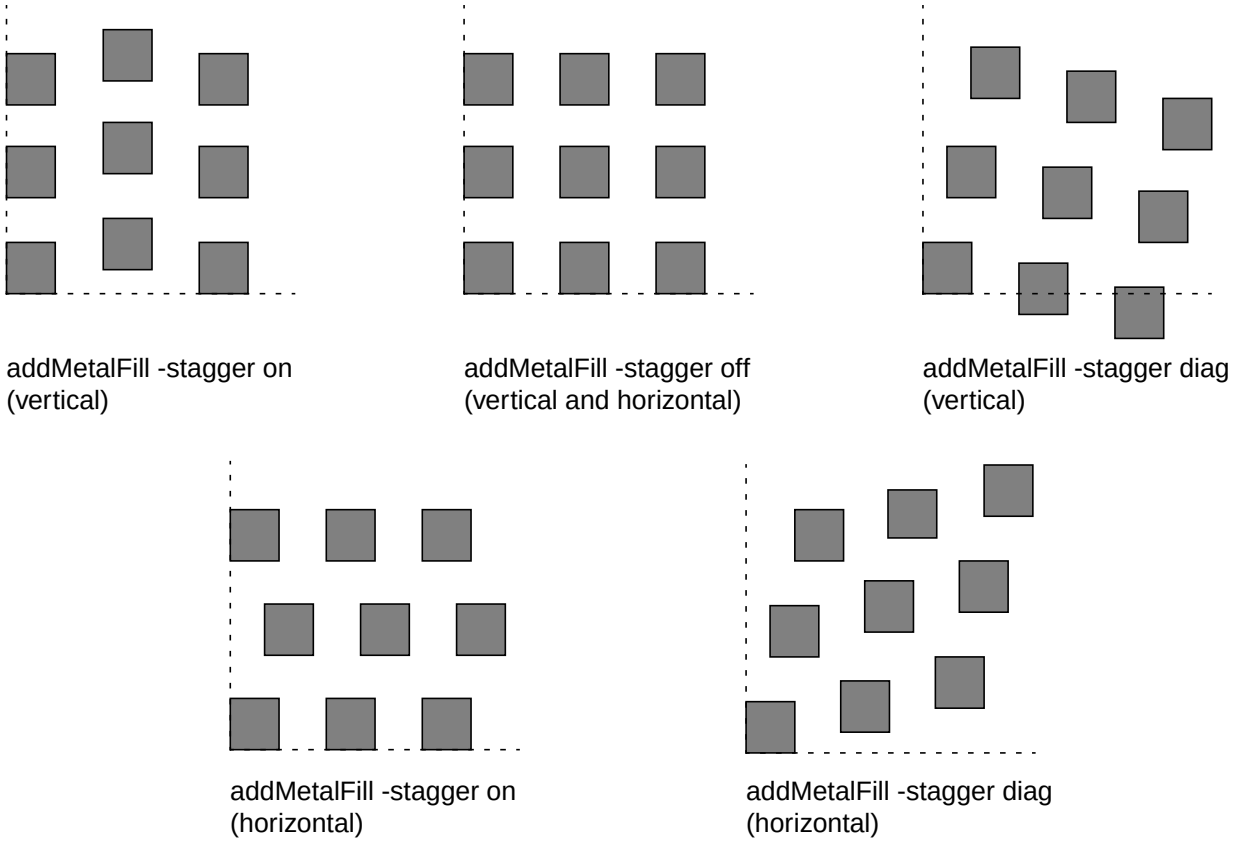
`off`

Adds metal fill shapes that line up in both the preferred and non-preferred routing directions, so that metal fill is not staggered.

`diag`

Adds metal fill shapes in a staggered pattern in both the preferred and non-preferred directions. Specify the offset for staggering with the following command: `setMetalFill -diagOffset` *`x y`*.

Figure 15-1 addMetalFill -stagger parameters



`-timingAware {on | off | sta}`

Specifies whether the software considers the timing impact of adding metal fill.
Default: `on`

Specify one of the following parameters:

`off`

Does not consider timing.

on

Assigns timing costs to nets and adds metal fill to nets with the lowest timing cost first. Adds less metal fill near clock and signal routing.

Divides nets into categories based on timing cost. The categories are:

- Clock nets (highest cost)
- Signal nets (moderate cost)
- Power and ground nets—nets marked + `USE POWER` or + `USE GROUND` in the DEF file (0 cost)

`sta`

Runs in timing-aware mode using the Encounter common timing engine (CTE) for static timing analysis (STA). To benefit from using the `sta` parameter, you must have run timing analysis with the `report_timing` command prior to adding metal fill. If there are no timing structures in memory, the `sta` parameter behaves like the `on` parameter.

The `sta` parameter also divides nets into categories, but it uses the timing analysis results for assigning timing costs to nets. It adjusts the costs as a function of the slack (the worst slack has the highest cost) and uses the `slackThreshold` to separate critical and noncritical signal nets. Critical nets are signal nets with slack less than the threshold, and therefore have a higher cost than noncritical nets.

The categories are:

- Clock nets (highest cost)
- Critical signal nets (moderate cost)
- Non-critical signal nets (small cost)
- Power and ground nets—marked `+ USE POWER` or `+ USE GROUND` in the DEF file (0 cost)

Specify a value for `-slackThreshold` if you specify `sta`.

Example

The following command adds connected metal fill to the specified area on *metal1*, *metal2*, and *metal3*, using the values set by the `setMetalFill` command:

```
addMetalFill -layer {1 2 3} -area 100 200 300 400 -nets {VDD GND}
```

Related Topics

- Optimizing Metal Density chapter of the *Encounter User Guide*

- "Overview"
- "Staggered Metal Fill Pattern"
- "Connected and Floating Metal Fill"
- "Timing-Aware Metal Fill"
- "Recommendations for Adding Timing-Aware Metal Fill"
- "Adding Metal Fill Over Macros"
- "Recommendations for Power-Strapping Mode"

- metal_fill.tcl in the *Encounter Flat Implementation Flow Guide*
- metal_fill.tcl in the *Encounter Hierarchical Implementation Flow Guide*

addViaFill

```
addViaFill
     [-area x1 y1 x2 y2]
     [-layer {cutLayerNumber | {list_of_cut_layer_numbers}]
     [-mode {all | floatingOnly | connectedOnly}]
     [-snap]
```

Uses the values specified by the `setMetalFill` command to determine sizes of vias to create. Only vias that meet Minimum Area Rule requirements are used. It also uses values from `setViaFill` for the window size, step, and density values.

For information, see "setMetalFill" on page 427 and "setViaFill" on page 435.

Note: Default values from `setMetalFill` for `minWidth` and `minLength` are biased towards the needs of `addMetalFill` use and should be overridden with values that are more useful for via generation.

Add via fill in the following areas immediately before adding metal fill:

- Where metal fill shapes would intersect with power or ground nets on consecutive layers if no via fill was added.
- Where metal fill shapes on consecutive layers would intersect if no via fill was added.

Tip

To reduce the need for via fill, insert multiple-cut vias with the NanoRoute router. For information, see setNanoRouteMode on page 1280.

Parameters

`-area x1 y1 x2 y2`

Specifies a set of coordinates within which to add via fill.
Default: If you do not specify this parameter, adds via fill to the entire design area.

`-layer {`*`cutLayerNumber`*`|{`*`list_of_cut_layer_numbers`*`}}`

Specifies the cut layers for adding via fill. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks. To specify *cut12*, *cut23*, and *cut34*, type one of the following commands:

```
addViaFill -layer "V12 V23 V34"
addViaFill -layer {V12 V23 V34}
```

Default: If you do not specify this parameter, adds via fill to all cut layers.

`-mode {all | floatingOnly | connectedOnly}`

Specifies the connection mode for via fill. Via fill can be connected to power or ground nets (tied off) or unconnected (floating). Via fill cannot be connected to signal nets.
Default: `all`

Specify one of the following parameters:

`all`

Enables addition of both connected and unconnected via fill.

`floatingOnly`

Adds unconnected via fill only.

`connectedOnly`

Adds connected via fill only.

`-snap`

Snaps edges of via fill shapes and centers of cuts to a user-defined snap grid that is a multiple of the manufacturing grid. When via shapes are snapped to this grid, the edges and centers of the via shapes lie on the grid. If a grid is not specified, the software snaps the shapes to the manufacturing grid. You can define a grid using the `setSnapGrid` command.

For more information, see setSnapGrid on page 2261.

Related Topics

- Optimizing Metal Density chapter of the *Encounter User Guide*

deleteMetalFill

```
deleteMetalFill
     [-help]
     [-area x1 y1 x2 y2]
     [-layer {layerNumber | {list_of_layer_numbers} } ]
     [-mode {all | connectedOnly | floatingOnly} ]
     [-shapes {[FILLWIRE] [FILLWIREOPC}] ]
```

Removes metal and via fill in order to start over with the metal fill process. If you run this command without specifying parameters, it deletes all the metal and via fill in the design.

Note: Deleting via fill with this command might create open violations.

Parameters

`-area` *x1 y1 x2 y2*

Specifies a set of coordinates within which to delete metal and via fill.
Default: If you do not specify this parameter, deletes fill from the entire design area .

`-help`

Outputs a brief description that includes type and default information for each `deleteMetalFill` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man deleteMetalFill`.

`-layer {`*layerNumber* | **{***list_of_layer_numbers***}** `}`

Specifies the layers from which to remove metal and via fill. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks. This parameter accepts metal layers, cut layers, or a combination.

- If you specify a cut layer, the command removes vias that contain that cut layer only.
- If you specify a metal layer, metal fill shapes on that layer and vias that contain that layer are removed.

Default: If you do not specify this parameter, removes metal and via fill from all layers.

`-mode {all | connectedOnly | floatingOnly}`

Specifies whether the command removes only connected fill, only floating fill, or all fill.
Default: `all`

`all` Removes connected and floating metal and via fill.

`connectedOnly`
Removes only connected metal and via fill.

`floatingOnly`
Removes only floating metal and via fill.

`-shapes {[FILLWIRE] [FILLWIREOPC]}`

Specifies whether the software removes `FILLWIRE` or `FILLWIREOPC` shapes or both kinds of shapes.
Default: `FILLWIRE FILLWIREOPC`

Example

The following command removes `FILLWIRE` via fill from cut layers *V12*, *V23*, *V34*, *V45*, *V56*:

```
deleteMetalFill -layer {V12 V23 V34 V45 V56}
```

setMetalFill

```
setMetalFill
     -layer {layerNumber | {list_of_layer_numbers}
     [-iterationName name]
     [-opc]
     [-maxLength length]
     [-minLength length]
     [-maxWidth width]
     [-minWidth width]
     [-decrement value]
     [-activeSpacing spacing]
     [-externalDensity percent]
     [-gapSpacing spacing]
     [-windowSize xSize ySize]
     [-windowStep xStep yStep]
     [-minDensity percent]
     [-maxDensity percent]
     [-preferredDensity percent]
     [-diagOffset x y]
```

Specifies values for parameters that the `addMetalFill`, `trimMetalFill`, and `verifyMetalDensity` commands use for metal fill size, spacing, verification, and metal density. You can also specify some of these parameters in the Layer (Routing) section of the LEF file – they are identified in the "Parameters" section below.

Note: Check the chip manufacturer's DRC manual specifications for metal fill (sometimes called dummy fill) for the correct values if they are not specified in the LEF file.

- For more information on LEF, see the "LEF Syntax" chapter in the *LEF/DEF Language Reference*.
- For more information on the metal fill commands, see addMetalFill on page 410, trimMetalFill on page 437, and verifyMetalDensity on page 2284.

Parameters

`-activeSpacing` *spacing*

Specifies the minimum distance in microns between a metal fill shape and any other type of object in the design.

Note: The `setMetalFill -activeSpacing` settings are used for creating regular `FILLWIRE` shapes. For `FILLWIREOPC` shapes, design rules are used.

LEF equivalent: `FILLACTIVESPACING`
Default: `FILLACTIVESPACING` in the LEF file; if not specified, `0.6` for thin layers (less than 0.24), `0.8` for thick layers

`-decrement` *value*

Specifies the number of microns the software decreases the size of each successively smaller piece of metal fill or via fill until it reaches the size specified by the `-minWidth` parameter.

The software uses the maximum fill size specified unless it is impossible to fit a piece of fill into a particular area. The software then attempts to use successively smaller pieces of fill until the minimum length value is reached.

This parameter allows you to control how many different sizes of metal fill or via fill can appear within the design.

For example, if the value of `-maxWidth` is `8`, the value of `-minWidth` is `4`, and the value of `-decrement` is `2`, the software can add fill shapes with widths of `8`, `6`, and `4` microns.

The software tries to add fill shapes with the width specified by `-maxWidth`. If a shape of that width does not fit in the available space without creating a violation, the software decreases the width by the amount specified by the `-decrement` parameter. It continues to decrease the width until it reaches the width specified by the `-minWidth` parameter.

To increase flexibility in adding metal or via fill, decrease the value of this parameter. For example, if the value of `-decrement` is `1`, metal fill with widths of `8`, `7`, `6`, `5`, and `4` microns can be added.

Default: `WIDTH` value specified in the `Layer` (Routing) statement of the LEF file

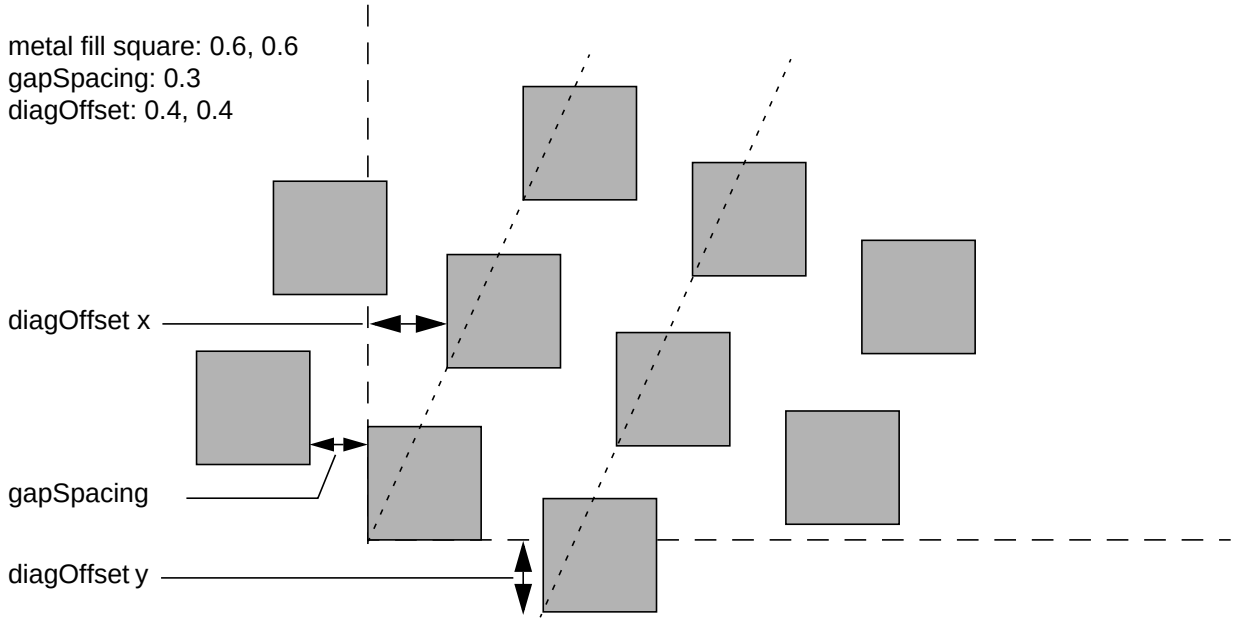
`-diagOffset` *x y*

Specifies the offset for staggering metal fill. To enable this parameter, `addMetalFill -stagger diag` must be specified.

x Specifies the horizontal offset.

y Specifies the vertical offset.

Figure 15-2 setMetalFill diagOffset *x y*



`-externalDensity` *percent*

Specifies a percentage value for the external metal density for a layer when `addMetalFill` windows lie partially outside of the die area. For each layer, a different external density value can be specified.

Metal fill uses the external density value in the estimation of layer density outside of the blocks being developed.

`-gapSpacing` *spacing*

Specifies the minimum distance in microns between one metal fill shape and another metal fill shape.
Default: `0.4` for thin layers (less than 0.24 width/minWidth), `0.8` for thick layers

`-iterationName` *`name`*

Specifies a name for a set of values for `setMetalFill` parameters that the software stores and uses during a later `addMetalFill` run. Use `addMetalFill -iterationNameList` to add the metal fill using the stored set of parameters.

The window size and step must be the same for all iterations of a specific layer. For example, the following specifications are not allowed because the values are not consistent:

```
setMetalFill -iterationName step1 -layer 1 \
  -minDensity 20 -windowSize 100 100 -windowStep 50 50
setMetalFill -iterationName step1 -layer 1 \
  -minDensity 25 -windowSize 50 50 -windowStep 25 25
```

Important

Specifying `setMetalFill -iterationName default` is the same as running `setMetalFill` without specifying the `-iterationName` parameter.

`-layer` {*`layerNumber`* | **{***`list_of_layer_numbers`***}**

Specifies the layers to which the parameters of the `setMetalFill` command apply. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks.

To specify metal layers 1, 2, and 3, type one of the following commands:

```
setMetalFill -layer "1 2 3"
setMetalFill -layer {1 2 3}
```

`-maxDensity` *`percent`*

Specifies a percentage value for the maximum metal density allowed.
LEF equivalent: `MAXIMUMDENSITY`
Default: `MAXIMUMDENSITY`. If this value is not defined, the default is `80`.

Note: This parameter only overrides a value in the LEF file if it is more restrictive than the LEF value. For example, if the value specified in the LEF file is 80, and you specify `-maxDensity 70`, metal fill uses 70 percent as the maximum density.

`-maxLength` *`length`*

Specifies the maximum length of the metal fill shapes, in microns. Metal fill stops at the chip boundary if the specified maximum length would cause it to extend outside the boundary.
Default: `10.0`

`-maxWidth` *`width`*

Specifies the maximum width of metal fill shapes, in microns. Metal fill stops at the chip boundary if the specified maximum length would cause it to extend outside the boundary.
Default: Maximum routing width defined in the LEF file; if not specified, `2.0`
LEF equivalent: `MAXWIDTH`

Note: This parameter only overrides a value in the LEF file if it is more restrictive than the LEF value.

`-minDensity` *`percent`*

Specifies a percentage value for the minimum metal density allowed in the design.
LEF equivalent: `MINIMUMDENSITY`
Default: `MINIMUMDENSITY`. If this value is not defined, the default is `20`.

Note: This parameter only overrides a value in the LEF file if it is more restrictive than the LEF value. For example, if the value specified in the LEF file is 20, and you specify `-minDensity 25`, metal fill uses 25 percent as the minimum density.

`-minLength` *`length`*

Specifies the minimum length of metal fill shapes, in microns.
Default: `1.0`

`-minWidth` *`width`*

Specifies the minimum width of metal fill shapes, in microns.
Default: `0.4` for thin layers (less than 0.24) and `0.8` for thick layers.

Note: This parameter only overrides a value in the LEF file if it is more restrictive than the LEF value.

`-opc`

Specifies metal fill that will go through the optical proximity correction (OPC) process. This fill can be smaller than non-OPC metal fill and is governed by the same design rules as regular wires, instead of the rules that govern non-OPC metal fill.

- OPC metal fill that is floating (unconnected) is included in the Fills statement of DEF 5.7.
- OPC metal fill that is tied off (connected) is included in the Special Wiring statement of DEF 5.7.
- The GDSII Stream/OASIS map file identifies the following object names for OPC and non-OPC metal fill: `FILL`, `FILLOPC`, `VIA`, `VIAFILL`.

`-preferredDensity` *`percent`*

Specifies a percentage value for the preferred metal density for the layer. When the metal density in a window is less than the minimum metal fill density value, adds metal fill to achieve a density slightly above the preferred density if possible.

In addition, metal fill uses the prefered density value in the estimation of `MACRO` density for layers that have most of the `MACRO` covered by `OBS` shapes and for DEF `BLOCKAGES` without both `+ FILLS` and `+ PUSHDOWN`.

Metal fill uses the following scheme for the treatment of DEF blockages with `+ FILLS` or `+ PUSHDOWN`:

- DEF `BLOCKAGES` with `+ PUSHDOWN` = 100 percent density (the same as real metal segments)
- DEF `BLOCKAGES` with `+ FILLS` = 0 percent density

Cadence recommends modelling LEF `MACRO`s by using the `DENSITY` statement, with density windows that are smaller than the metal fill windows. For example, if the metal fill windows are 100 x 100, use 25 x 25 or smaller values for the density rectangles.

Default: 35

`-windowSize` *`xSize ySize`*

Specifies the x and y dimensions in microns of the area `setMetalFill` uses to examine the metal density. The software assesses the metal density in each window, moving the window iteratively through the design.
LEF equivalent: `DENSITYCHECKWINDOW`
Default: `DENSITYCHECKWINDOW`. If this value is not defined, the default is 100 microns in each dimension.

`-windowStep` *`xStep yStep`*

Specifies the x and y step distance in microns the window moves for each iteration of metal density analysis.
LEF equivalent: `DENSITYCHECKSTEP`
Default: `DENSITYCHECKSTEP`. If this value is not defined, the default is 50 microns in each dimension.

Example

The following command sets metal fill parameters for layer `1`:

```
setMetalFill -layer 1 -windowSize 200 200 -windowStep 100 100
```

These parameters are applied when you use the `addMetalFill` command.

Related Topics

- Optimizing Metal Density chapter in the *Encounter User Guide*
  - "Overview"
  - "Connected and Floating Metal Fill"
  - "Specifying Metal Fill Parameters"
  - "Timing-Aware Examples"
  - "Specifying the Active Spacing Value"
  - "Adding Via Fill"
- Importing and Exporting Designs chapter in the *Encounter User Guide*
  - "GDSII Stream or OASIS Map File Format"
- LEF Syntax chapter in the *LEF/DEF Language Reference*

- "Macro Density"

- DEF Syntax chapter in the *LEF/DEF Language Reference*
  - "Blockages"
- metal_fill.tcl in the *Encounter Flat Implementation Flow Guide*

setViaFill

```
setViaFill
     -layer {layerNumber | {list_of_layer_numbers}}
     [-windowSize xSize ySize]
     [-windowStep xStep yStep]
     [-minDensity percent]
     [-maxDensity percent]
     [-preferredDensity percent]
```

Specifies values for parameters for addViaFill.

Parameters

`-layer` *`layerNumber`* | **{***`list_of_layer_numbers`***}**

Specifies the cut layers to which the parameters of the `setViaFill` command apply. If you specify more than one layer, separate each layer with a space and enclose the list in braces or double quotation marks. To specify *cut12*, *cut23*, and *cut34*, type one of the following commands:

```
addViaFill -layer "V12 V23 V34"
addViaFill -layer {V12 V23 V34}
```

`-maxDensity` *`percent`*

Specifies a percentage value for the maximum cut density.
Default: `30`

`-minDensity` *`percent`*

Specifies a percentage value for the minimum cut density.
Default: area of two cuts divided by 100 square microns

`-preferredDensity` *`percent`*

Specifies a percentage value for the preferred cut density. When the cut density in a window is less than the minimum cut density value, this parameter adds via fill to achieve a density as close as possible to the preferred value. To disable the preferred density feature, specify `0`.
Default: area of two cuts in the cut layer

`-windowSize` *`xSize ySize`*

Specifies the x and y dimensions in microns of the area `setViaFill` uses to examine the cut density. The software assesses the cut density in each window, moving the window iteratively through the design.
Default: 10 microns in each dimension

`-windowStep` *`xStep yStep`*

Specifies the x and y step distance in microns the window moves for each iteration of cut density analysis.
Default: 5 microns in each dimension

Related Topics

- Optimizing Metal Density chapter of the *Encounter User Guide*
  - "Overview"
  - "Adding Via Fill"

trimMetalFill

```
trimMetalFill
     [-allowPolygons]
     [-deleteViols]
     [-ignoreSpecialNets]
```

Trims metal fill and via fill (`FILLWIRE` and `FILLWIREOPC` shapes) that causes DRC violations. This command uses the values set by the `setMetalFill` command.

Use this command to trim metal fill if the design has floating metal or via fill. By trimming the fill, you reduce the chance of introducing timing problems caused by cross-coupling.

If the design has metal or via fill that is connected to special nets, delete the fill and re-insert it instead of using this command. To delete metal fill, use the `editDelete -shapes FILLWIRE` command.

Use this command after routing signal nets. For more information, see "Route Commands" on page 1223.

Parameters

`-allowPolygons`

Allows rectilinear shapes after trimming metal fill (`FILLWIRE` and `FILLWIREOPC` shapes).

By default, the Encounter software supports rectangular shapes only.

`-deleteViols`

Deletes metal fill shapes that cause DRC violations or shorts, instead of trimming sections of the shapes. After running the `trimMetalFill` command with this parameter specified, the remaining shapes are still rectangles. If using this parameter causes metal density to drop below the target density, re-run the `addMetalFill` command.

Always specify this parameter in the following two cases:

- If the design has minimum step rules, specify this parameter to avoid creating minimum step violations.
- If you ran `addMetalFill -snap`, specify this parameter to delete entire shapes, instead of trimming sections of the shapes, otherwise the trimmed shapes will not be on-grid.

`-ignoreSpecialNets`

Ignores special nets while checking for DRC violations.

Example

The following commands trim metal fill after ECO routing.

```
setNanoRouteMode -routeWithEco true
globalDetailRouteBatch
trimMetalFill
```

Related Topics

- Optimizing Metal Density chapter of the *Encounter User Guide*
  - "Overview"
  - "Connected and Floating Metal Fill"
  - "Trimming Metal Fill"
- DEF Syntax chapter of the *LEF/DEF Language Reference*
  - "Shape"

16

Mixed Signal Commands

createConstraintFileFromDB

`createConstraintFileFromDB`

Creates a constraint file for the mixed signal router and verifier from the Encounter database. The file is a text file with the following extension: `.const`.

Parameters

None

deleteAllMsConstraints

`deleteAllMsConstraints`

Deletes mixed signal constraints from the Encounter database.

Parameters

None

readConstraintFileInDB

`readConstraintFileInDB` *`filename`*

Reads a constraint file created by the `createConstraintFileFromDB` command into the Encounter database. The mixed signal router and verifier use this file to constrain routing and run verification checks.

Parameters

`filename`	Specifies the file to read.

reportAnalog

```
reportAnalog
     [-all | -inst | -net]
     [-highlight | -dehighlight]
     [-reportFile fileName]
```

Generates a report that lists analog objects in the Encounter database. Optionally, highlights the objects in the design display area of the main Encounter window or removes the highlights.

If you run this command without parameters, it generates a report of analog nets and instances.

Use `setAnalog` to mark or unmark the objects as analog. For more information, see "setAnalog" on page 447.

Parameters

`-all`	Selects all objects marked as analog.
`-inst`	Selects all instances marked as analog.
`-net`	Selects nets marked as analog. Nets that are marked as analog, but are still unrouted, are also selected.
`-reportFile` *`fileName`*	Generates a report file that contains an alphabetical list of objects marked as analog. Lists unrouted analog nets separately from routed nets. *Default:* *`design_name`*`.analog.rpt`
`-highlight`	Highlights the specified type of analog objects or all analog objects.
`-dehighlight`	Removes highlights from the specified type of analog objects or from all analog objects.

routeMixedSignal

```
routeMixedSignal
     -constraintFile fileName
     -jogControl {preferDifferentLayer | preferSameLayer |-noLayerChange}
     [-area {llx lly urx ury}]
     [-deleteExistingRoutes]
     [-layerChangeBottomLayer layerNumber]
     [-layerChangeTopLayer layerNumber]
     [-nets {all | {netNames}}]
     [-noShareShield]
```

Routes digital and analog nets, honoring constraints for the following types of complex routing:

- Net shielding, including coaxial shielding, in which the router shields noise-sensitive nets on the top and bottom, in addition to the right and left sides
- Matched routing, in which the router attempts to create routes that are the same length, although they are not required to be topologically similar
- Differential routing, in which the router attempts to match the parasitic load on driving pins of nets
- Bus routing, in which the router routes “bundles” of nets that follow the same pattern from the source to the target

Parameters

`-area` Specifies the area to route.

`-constraintFile` *`fileName`*

Specifies the file that contains constraints for differential pair, matched pair, and shielded routing, and for routing with special width and resistance constraints.

`-deleteExistingRoutes`

Removes existing connections when you use this command multiple times.
Default: Maintains existing connections

Note: If you specify the `-area` parameter, existing routes are always preserved, even if you also specify this parameter.

`-jogControl {preferDifferentLayer | preferSameLayer | -noLayerChange}`

Allows jogs during routing to avoid DRC violations.

Specify one of the following parameters:

`noLayerChange`

If the route must jog to avoid a DRC violation, the jog occurs on the same layer.

`preferSameLayer`

If the route must jog to avoid a DRC violation, the jog occurs on the same layer whenever possible. This can result in routing in the non-preferred direction.

`preferDifferentLayer`

If the route must jog to avoid a DRC violation, the jog occurs on the layer that is in the preferred routing direction whenever possible.

`-layerChangeBottomLayer` *`layerNumber`*

Specifies the bottom-most metal layer the mixed signal router uses.
Default: If you do not specify this parameter, all layers are used.

`-layerChangeTopLayer` *`layerNumber`*

Specifies the top-most metal layer the mixed signal router uses.
Default: The layer specified by `setMaxRouteLayer`. If no value is specified for `setMaxRouteLayer`, uses the top-most routing layer as the default value.

`-nets {all | {`*`list_of_net_names`*`}}`

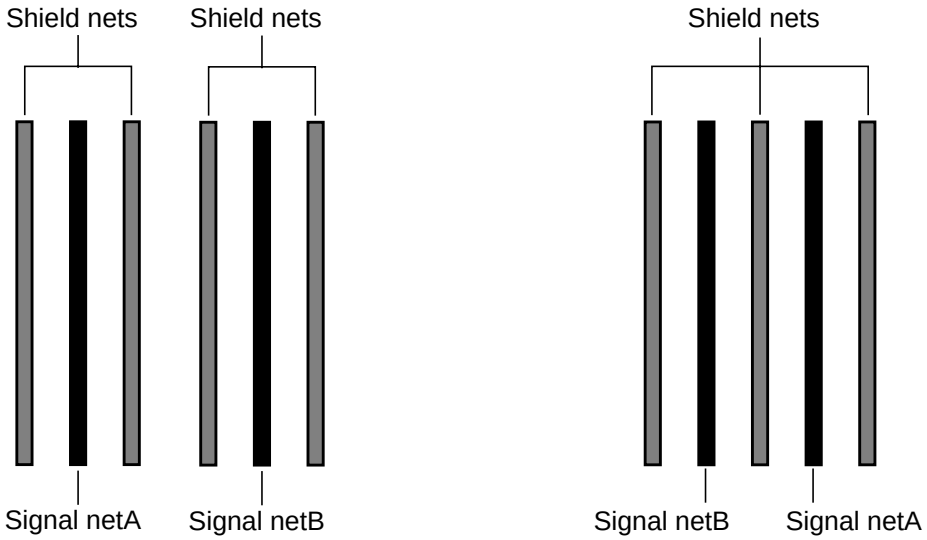
Specifies the nets to route. If you specify more than one net name, enclose the names in braces and separate the names with a space, in the following format:

```
-nets {netA netB netC}
```

Default: `all`

`-noShareShield`

Does not allow shielded nets to share a shield net when the nets are parallel. The following figures show two nets that are shielded on the right and left sides. In the figure on the left, signal netA and signal netB each has its own shield nets. In the figure on the right, signal netA and signal netB share the center shield net.



Related Topics

- Using the Encounter Mixed Signal Router chapter of the *Encounter User Guide*
- Verifying Violations chapter of the *Encounter User Guide*
  - "Viewing Violations With the Violation Browser"

setAnalog

```
setAnalog
     {-allCellNets cellName |
      -analogTerms |
      -cell cellName |
      -inst instanceNames |
      -instsAndNets instanceNames |
      -msConsts {all | bus | diffPair | matchPair | shielded} |
      -net signalNetNames |
     [-unset]}
```

Adds an analog attribute to specified database objects or removes the attribute if it was added previously with this command. Adding this attribute allows the Encounter software to treat the objects differently than it treats digital objects.

Use `reportAnalog` to highlight the objects in the main Encounter window and generate a report. For more information, see "reportAnalog" on page 443.

Important

The NanoRoute router routes analog nets unless you specify the following command before routing:

```
setNanoRouteMode -dbSkipAnalog true
```

Parameters

`-allCellNets cellName`

Marks or unmarks as analog all instances of all nets that interface with the specified cell. Use this parameter when marking the nets manually is not practical due to the large number of nets to mark.

`-analogTerms` Marks or unmarks as analog all nets connected to terminals marked `dbcAnalogTerm`.

`-cell cellName` Marks or unmarks as analog all instances of the specified cell.

`-inst instanceNames`

Marks or unmarks as analog the specified instance or list of instances.

`-instsAndNets` *`instanceNames`*

Marks or unmarks as analog all nets that interfaces with the specified instance or list of instances. Use this parameter when marking the nets manually is not practical due to the large number of nets to mark.

`-msConst {all | diffPair | matchPair | shielded | bus}`

Marks or unmarks as analog all nets on which the specified type of mixed signal constraint is applied.

Specify one of the following options:

`all`	Marks or unmarks all nets with any mixed signal constraints.
`bus`	Marks or unmarks all nets with bus constraints.
`diffPair`	Marks or unmarks all nets with differential pair constraints.
`matchPair`	Marks or unmarks all nets with matched net constraints.
`shielded`	Marks or unmarks all nets with shielding constraints.

`-net` *`signalNetNames`*

Marks or unmarks as analog the specified signal net or a list of signal nets.

`-unset` Resets objects that were set to analog as digital objects.

Examples

- The following command marks as analog all nets on which matchPair constraints are applied:

  ```
  setAnalog -msConsts matchPair
  ```

- The following command resets `net_a`, `net_c`, and `net_f` to digital objects:

  ```
  setAnalog -unset -net "net_a net_c net_f"
  ```

Related Topics

- Using the Encounter Mixed Signal Router chapter of the *Encounter User Guide*.

- “Specialized Constraints and Keyword Descriptions”

verifyAnalogRoutingConstraints

```
verifyAnalogRoutingConstraints
     [-nets {netName | list_of_nets}]
     [-diffPair]
     [-dontCross]
     [-filterGapH value]
     [-filterGapL value]
     [-filterResH value]
     [-filterResL value]
     [-filterWidthH value]
     [-filterWidthL value]
     [-matchedNets]
     [-report fileName]
     [-routedNets]
     [-shieldedNets]
     [-bus]
```

Checks that nets routed by the mixed signal router meet the constraints specified by the mixed signal routing constraint editor. Generates a violation report. You can limit the violations reported by specifying high and low values for spacing violations, resistance, and width to report.

The verifier checks that net resistance, capacitance, width, and minimum via cut constraints are met. In addition, it checks the following types of nets:

- Differential pair nets

  The parasitic load on driving pins of the nets must be equal.

- Matched nets

  Nets must be the same length.

- Routed nets

  Nets must meed width, resistance, capacitance, and minimum via cut constraints.

- Shielded nets

  Noise-critical nets are shielded by special nets on their right and left sides, or on the right, left, top, and bottom sides.

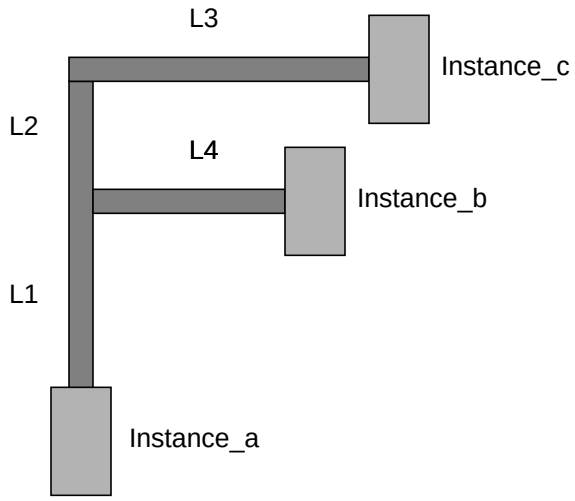
- Do not cross nets

  Potentially noise-causing nets are prohibited from crossing noise-sensitive nets.

Maximum and Minimum Resistance Constraints

Nets must meet maximum and minimum resistance requirements. The verifier marks a violation on the output pin of the instance or block driving the net if the maximum resistance is greater than the maximum resistance specified by the constraint or if the minimum resistance is less than the minimum resistance specified by the constraint.

The following figure shows a multi-pin net with four segments.



The verifier calculates effective resistance using the following formula:

Resistance =
((Resistivity of L1)*(L1/Width of L1)) +
((Resistivity of L2)*(L2/Width of L2)) +
((Resistivity of L3)*(L3/Width of L3))

- If the pin on `Instance_b` is an output pin, valid paths are `L1 + L2 + L3` and `L1 + L4`.
- If the pin on `Instance_b` is a bidirectional pin, valid paths are `L1 + L2 + L3`, `L1 + L4`, and `L4 + L2 + L3`.

Maximum Capacitance Constraints

Nets must meet maximum capacitance requirements. The verifier marks a violation on the output pin of the instance or block driving the net if the capacitance exceeds the maximum resistance specified by the constraint.

The software calculates capacitance using the following formula:

Total capacitance =
area_capacitance + side capacitance

Minimum Cuts Constraints

Vias must not have less than the specified minimum number of cuts.

For more information on the mixed signal constraints, see "Mixed Signal Constraint Editor" in the *Encounter Menu Reference* or Using the Encounter Mixed Signal Router in the *Encounter User Guide*.

Parameters

`-bus`	Checks that bus constraints, in which specified nets must be routed with the same width and spacing, are met.
`-diffPair`	Checks that constraints on differential pair routing, in which the router attempts to match the parasitic load on driving pins of nets, are met.
`-dontCross`	Checks that the specified noise-generating nets do not cross specified nets that are noise sensitive.
`-filterGapH` *`value`*	Does not report spacing violations that are greater than the specified value.
`-filterGapL` *`value`*	Does not report spacing violations that are less than the specified value.
`-filterResH` *`value`*	Does not report resistance violations that are greater than the specified value. Uses the same resistance formula the mixed signal constraint editor and router use. For more information, see *Max Res*.
`-filterResL` *`value`*	Does not report resistance violations that are less than the specified value. Uses the same resistance formula the mixed signal constraint editor and router use. For more information, see *Max Res*.

`-filterWidthH` *`value`*

Does not report width violations that are greater than the specified value.

`-filterWidthL` *`value`*

Does not report width violations that are less than the specified value.

`-matchedNets` Checks constraints on matching routing, in which the router attempts to create routes that are the same length. Matching routing is not limited to two nets, and the nets are not required to be topologically the same.

`-nets {`*`netName | list_of_nets`*`}`

Specifies the net, or list of nets, to check.

`-report` Specifies the report name.
Default: *`designName`*`.constraints.rpt`.

`-routedNets` Checks that other nets routed by the mixed signal router meet constraints on width, resistance, and capacitance.

`-shieldedNets` Checks shielded routing, including regular shielding in which noise-critical nets are shielded on their right and left sides, and coaxial shielding, in which nets are shielded on the right, left, top, and bottom sides.

Constraints for shielding include the names of the nets to shield (signal nets), the names of the shielding nets (power or ground nets), the width of the shielding net, and the spacing between the shielded nets and the shielding nets.

Related Topics

- Using the Encounter Mixed Signal Router chapter of the *Encounter User Guide*
- Verifying Violations chapter of the *Encounter User Guide*
  - "Viewing Violations With the Violation Browser"

verifyElectricalConstraints

```
verifyElectricalConstraints
     -IRDrop
     [-report designname.electricalconst.rpt]
```

Checks for IR drop violations that are greater than the maximum specified by the mixed signal constraints. Generates a violation report.

Parameters

`-IRDrop` Checks for IR drop violations.

Important

Before you check the IR drop, you must run `updatePower` to set the actual value for instance pin IR drop. Otherwise, all the IR drop values with instance pin are 0, and you will not see any IR drop violations.

`-report designname.electricalconst.rpt`

Specifies the report generated by this command.

17

Multiple-CPU Processing Commands

getDistributeHost

```
getDistributeHost
     {-mode | -hosts | -queue | -resource | -custom_script | -lsf_args}
     [-help]
```

Displays settings for the setDistributeHost command.

Parameters

`-custom_script`

Displays the name of the custom script when the software is running in `custom` mode.

`-hosts` Displays the list of host names when the software is running in `rsh` mode.

`-help` Outputs the command usage and a brief description about the command parameters.

`-lsf_args`

Displays the lsf arguments when the software is running in `lsf` mode.

`-mode` Displays the distributed processing mode. The possible values are `local`, `sge`, `rsh`, `lsf`, `custom`, or `0`. The command displays `0` when the software is not in multiple-CPU processing mode.

`-queue` Displays the queue name when the software is running in `SGE` or `lsf` mode.

`-resource`

Displays the resource string when the software is running in `lsf` mode.

Related Topics

- Accelerating the Design Process by Using Multiple-CPU Processing chapter in the *Encounter User Guide*
- Design Menu chapter in the *Encounter Menu Reference*
  - "Multiple-CPU Processing"

getMultiCpuLicense

```
getMultiCpuLicense
     [numCPUs]
     [ [-nru][-socel][-socexl][-socegxl] [-nru] [-edsl] [-edsxl] ]
```

Specifies the number of CPUs requested, the licenses to check out, and the license search order. After the licenses are checked out, they can be used for multi-threading, distributed processing, or superthreading.

For detailed information on multi-CPU licensing, see Encounter Digital Implementation System Licensing and Packaging and SoC Encounter Licensing and Packaging on SourceLink®.

Important

This command is optional. If you do not run it, the software runs it automatically, using the default search order, and checks out the necessary number of licenses based on the parameters you set for `setMultiCpuUsage`.

Parameters

`[-nru][-socel][-socexl][-socegxl] [-nru] [-edsl] [-edsxl]`

Specifies the licenses to check out and the search order. The software checks out all possible copies of the first option you list, then the second option, and so on, until the requested number of multiple-CPU licenses is reached. If enough licenses are not available, the software issues a warning.

If you started the software using the `encounter` command, choose from the following options:

- `-nru`
- `-socel`
- `-socexl`
- `-socegxl`

If you started the software using the `velocity` command, choose from the following options:

- `-nru`
- `-edsl`
- `-edsxl`

Note: For descriptions of each of the options, see Product and Licensing Information in the *Encounter User Guide*.

`numCPUs`

Specifies the number of CPUs required. For example, if you specify `getMultiCpuLicense 10`, the software checks out enough licenses to enable 10 CPUs.

Related Topics

- Accelerating the Design Process by Using Multiple-CPU Processing chapter in the *Encounter User Guide*
- Placing the Design chapter in the *Encounter User Guide*

- “Running Placement in Multi-CPU Mode”

- Design Menu chapter in the *Encounter Menu Reference*
  - “Multiple-CPU Processing”

getMultiCpuUsage

```
getMultiCpuUsage
     {-numHosts | -numThreads | -superThreadsNumHosts -superThreadsNumThreads}
```

Displays the requested number of threads or hosts.

To request threads or hosts, use `setMultiCpuUsage`.

Parameters

`-numHosts`	Displays the requested number of hosts for distributed processing.
`-numThreads`	Displays the requested number of threads for multi-threading.
`-superThreadsNumHosts`	Displays the requested number of hosts for Superthreading.
`-superThreadsNumThreads`	Displays the requested number of threads for Superthreading.

Examples

The following command displays the requested number of threads for multi-threading:

```
getMultiCpuUsage -numThreads
```

The following command displays the requested number of hosts and threads for Superthreading:

```
getMultiCpuUsage -superThreadsNumHosts -superThreadsNumThreads
```

Related Topics

- Accelerating the Design Process by Using Multiple-CPU Processing chapter in the *Encounter User Guide*

getReleaseMultiCpuLicense

```
getReleaseMultiCpuLicense {true | false}
```

Queries the current setting for the `setReleaseMultiCpuLicense` command.

Parameters

`false`	`setReleaseMultiCpuLicense` is `false`.
`true`	`setReleaseMultiCpuLicense` is `true`.

Related Topics

- Accelerating the Design Process by Using Multiple-CPU Processing chapter in the *Encounter User Guide*

releaseMultiCpuLicense

`releaseMultiCpuLicense`

Releases all multiple-CPU licenses immediately. By default, the software holds multiple-CPU licenses until the end of the current session.

To specify that the software release multiple-CPU licenses after every multiple-CPU command runs, use `setReleaseMultiCpuLicense`.

Parameters

None

Related Topics

- Accelerating the Design Process by Using Multiple-CPU Processing chapter in the *Encounter User Guide*
- "Placing the Design" chapter in the *Encounter User Guide*
  - "Running Placement in Multi-CPU Mode"

reportMultiCpuLicense

`reportMultiCpuLicense`

Returns the following information:

- Number of CPUs requested by `setMultiCpuUsage`
- Number of CPUs in use
- Number of CPUs checked out

Parameters

None

Related Topics

- Accelerating the Design Process by Using Multiple-CPU Processing chapter in the *Encounter User Guide*

setDistributeHost

```
setDistributeHost
     {-rsh
       {-add {hostName1 hostName2 …} |
        -remove {hostName1 hostName2 …} |
        all} [-rshTimeout seconds] |
      -lsf
       [-queue queue_name]
       [-resource resource_string]
       [-lsf_args lsf_arguments] |
      -sge [-queue queueName] |
      -local |
      -custom -custom_script script
     }
```

Specifies the multiple-CPU processing configuration for distributed processing or Superthreading. The software finds hosts from the last `setDistributeHost` command specified before a distributed processing or Superthreading command starts.

Use `getDistributeHost` to display the current settings for the `setDistributeHost` command.

Important

This command is required for distributed processing and Superthreading.

Parameters

`-add {`*`hostName1 hostName2 …`*`}`

Adds the specified hosts to the `rsh` configuration. If you specify more than one host, separate each host name with a space.

Each time you specify a host, the software uses it to run one slave process. To run more than one slave process on a host, specify it more than once. For example, if you have two machines that each has two CPUs and you want to use both CPUs in both machines, type a command like the following:

```
setDistributeHost -rsh -add {machine1 machine2 \
    machine1 machine2}
```

`-custom -custom_script` *`script`*

Specifies a custom script to run distributed processing.

`-local` Runs all distributed processing jobs on the master machine.

`-lsf` Specifies a Load Sharing Facility configuration. If you do not specify any parameters for `-lsf`, the software uses default settings.

Important

To use this parameter, LSF must already be set up (typically by specifying the `LSF_ENVDIR` and `LSF_SERVERDIR` environment variables) and `bsub` must be in your search path. Contact your LSF administrator for details.

`-lsf_args` *`lsf_arguments`*

Specifies the `bsub` options. For information, see the LSF documentation.

`-queue` *`queue_name`*

Specifies the queue for the LSF or SGE configuration.

`-remove {`*`hostName1 hostName2`* `… | all}`

Removes the specified hosts or all hosts from the `rsh` configuration. If you specify more than one host, separate each host name with a space.

`-resource` *`resource_string`*

Specifies a resource string for the LSF queue.

Important

The correct resource string is specific to your installation. Contact your LSF administrator for the appropriate parameters and values.

`-rsh` Specifies a remote shell configuration. For information, see the `rsh` documentation.

`-rshTimeout` *`seconds`*

Specifies the number of seconds the host machine waits for other machines to become available for multiple-CPU processing.
Default: 5

`-sge` Specifies a Sun Grid Engine configuration. For more information, see the SGE documentation.

Examples

rsh

The following command creates an `rsh` configuration and adds hosts `host10`, `host11`, and `host12`:

```
setDistributeHost -rsh -add {host10 host11 host12}
```

The following commands create an `rsh` configuration with `host1` and `host2` as remote hosts and then adds `host3` and `host4` to the configuration:

```
setDistributeHost -rsh -add {host1 host2}
setDistributeHost -rsh -add {host3 host4}
```

LSF

The following command uses an existing LSF environment, and relies on the general LSF set up by the system administrator. In this example, the name of the queue implies that all the machines have 4 Gb of memory.

```
setDistributeHost -lsf -queue mem4G
```

The following command schedules a LSF job named `bigQueue` to run on a Linux machine for a maximum of 30 minutes. No other jobs can run on the machine while `bigQueue` is running.

```
setDistributeHost -lsf -queue bigQueue -lsf_args {-x -W 30} \
   -resource {OSNAME==Linux}
```

The following command schedules a LSF job named `smallQueue` to run on Linux machines with minimum speeds of 2000 MHz.

```
setDistributeHost -lsf -queue smallQueue -lsf_args {-W 20} \
    -resource {OSNAME==Linux && SPEED>2000}
```

Note: Check with your LSF administrator for the correct values for the resource string, as the parameters and their values differ in each installation. For example, the parameter name for the speed of the machine might be `MHZ` instead of `SPEED` (as in the preceding example), and the units might be specified in gigahertz instead of megahertz.

Local

The following command runs all distributed processing jobs on the machine on which you are running the master version of the Encounter software:

```
setDistributeHost -local
```

Custom

The following command runs distributed processing with a custom script:

```
setDistributeHost -custom -customScript {bsub -q bigQueue -x -R "OS==Linux &&
memory>20000"}
```

Related Topics

- Accelerating the Design Process by Using Multiple-CPU Processing chapter in the *Encounter User Guide*
- Design Menu chapter in the *Encounter Menu Reference*
  - "Multiple-CPU Processing"
- Flat Implementation Flow chapter of the *Flat Implementation Flow Guide*
  - "Route the Design and Run Postroute Optimization"

setMultiCpuUsage

```
setMultiCpuUsage
     {-numHosts integer |
      -numThreads {integer | max}|
      -superThreadsNumHosts integer -superThreadsNumThreads {integer| max} }
     [-threadInfo {0 | 1 | 2}]
```

Specifies the number of threads to use for multi-threading, the maximum number of computers to use for distributed processing, or the maximum number of computers and the number of threads to use for Superthreading. Optionally, reports usage information.

Important

This command is required for multi-threading, distributed processing, and Superthreading.

Parameters

`-numHosts` *integer*

Specifies the maximum number of computers to use for distributed processing operations.

During distributed processing, the software uses the same number of threads on each computer—you cannot specify the number of threads to use on a machine-by-machine basis.

Default: `1`

`-numThreads {integer | max}`

Specifies the number of threads to use in one machine for multi-threading operations. Upon completion, the log file generated by each thread is appended to the main log file.

Note: The `-numThreads` parameter limits the number of threads running concurrently. Although the software can create additional threaded jobs during run time, depending on the application in use, only the number of threads specified with this parameter are run at a given time.

Specify `max` to ask for the maximum number of available threads.

Default: `1`

`-superThreadsNumHosts` *`integer`*

Specifies the maximum number of computers to use in superthreading operations. During superthreading, the software uses the same number of threads in each slave invocation of the tool—you cannot specify the number of threads to use on a machine-by-machine basis.

Default: `1`

`-superThreadsNumThreads {`*`integer`* `| max}`

Specifies the number of threads in each slave machine to use for superthreading operations.

Specify `max` to auto-detect the maximum number of available threads on the master and use this value for all slaves.

Default: `1`

`-threadInfo {0 | 1 | 2}`

Reports usage information.

Default: `0`

Specify one of the following values:

`0` Does not write messages to the log file.

`1` Reports starting and ending information for each thread and when all threaded jobs are complete.

For example,

```
Starting threaded job 1...
Starting threaded job 2...
Starting threaded job 3...
Ending threaded job 2.
Ending threaded job 1.
Ending threaded job 3.
All threaded jobs finished.
```

2

Reports the same information as for 1, plus the following additional timing details for each thread, for the parent, and for the combined totals of all the threads:

- Elapsed time
- Processor time
- System time
- Memory

For example,

```
Starting threaded job 1...
Starting threaded job 2...
Starting threaded job 3...
Ending threaded job 2 (1 elapsed sec, 0
processor sec, 0 system sec, 1.160M).
Ending threaded job 3 (2 elapsed sec, 0
processor sec, 0 system sec, 0.895M).
Ending threaded job 1 (10 elapsed sec, 0
processor sec, 0 system sec, 1.172M).
All threaded jobs finished (10 elapsed
sec: 0 processor sec (parent), 0 system
sec (parent), 0 processor sec (threads),
0 system sec (threads)).
```

Examples

To run global placement with four threads, specify the following commands:

```
setMultiCpuUsage -numThreads 4
placeDesign
```

To run global placement with the maximum number of available threads, specify the following commands:

```
setMultiCpuUsage -numThreads max
placeDesign
```

To run detailed routing with the NanoRoute router in Superthreading mode with two machines and six threads, using an LSF queue, specify the following commands:

```
setDistributeHost -lsf -queue myLSFqueue
setMultiCpuUsage -superThreadsNumThreads 6 -superThreadsNumHosts 2
detailRoute
```

Related Topics

- Accelerating the Design Process by Multiple-CPU Processing chapter in the *Encounter User Guide*
- Placing the Design chapter in the *Encounter User Guide*
  - "Running Placement in Multi-CPU Mode"
- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - "Accelerating Routing with Multi-Threading and Superthreading"
- Design Menu chapter in the *Encounter Menu Reference*
  - "Multiple-CPU Processing"
- metal_fill.tcl in the *Encounter Flat Implementation Flow Guide*
- mode.tcl in the *Encounter Flat Implementation Flow Guide*

setReleaseMultiCpuLicense

```
setReleaseMultiCpuLicense {true | false}
```

Specifies whether to release multiple-CPU licenses immediately after each command that uses them runs. By default (`false`), the software holds all multiple CPU licenses until the current session ends.

Specify this command before running any commands that require multiple-CPU applications.

To release all multiple-CPU licenses immediately, use `releaseMultiCpuLicense`.

Parameters

`false`	Does not release or check in multiple-CPU licenses when an application finishes.
`true`	Releases or checks in multiple-CPU licenses when an application finishes.

Related Topics

- Accelerating the Design Process by Using Multiple-CPU Processing chapter in the *Encounter User Guide*
- Placing the Design chapter in the *Encounter User Guide*
  - "Running Placement in Multi-CPU Mode"
- Design Menu chapter in the *Encounter Menu Reference*
  - "Multiple-CPU Processing"

18

Netlist-to-Netlist Command

runN2NOpt

```
runN2NOpt
     [-appendWLMFile fileName]
     [-autoPathAdjust]
     [-backEndReport FE_slack_report_file]
     [-bypassPrecheck]
     [-mode {auto | custom}]
     [-effort {low | medium | high}]
     [-frontEndReport RC_endpoint_report_file]
     [-group]
     [-opCond name]
     [-preserveClockNetsAll]
     [-preserveClockNetsNoData]
     [-preserveFixedInst]
     [-preserveHierPinsWithSDC]
     [-preserveSizingDifferentLibpins]
     [-preserveTristateNets]
     [-optFanout]
     [-cwlm]
     [-cwlmCellLimit value]
     [-cwlmLib fileName]
     [-cwlmSdc fileName]
     [-incFirst {low | medium | high}]
     [-inheritCwlm]
     [-noInheritCwlm]
     [-insertTieHiLo]
     [-noGroup]
     [-noWriteSDC]
     [-optFanout]
     [-percentageEndPoints percentage_number]
     [-rcVar {name_value_pairs}]
     [-rcAttr {name_value_pairs}]
     [-preloadInclude fileNames]
     [-postPlaceOpt]
     [-postcheck]
     [-postloadInclude fileNames]
     [-myScriptFile fileName]
     [-saveToDesignName name]
     [-inDir dirName]
     [-outDir dirName]
     [-report report_types]
     [-removeTempFiles]
     [-tieHiCell cellName]
     [-tieLoCell cellName]
     [-multiVtEffort {medium | high | low }]
     [-maxLeakagePower value]
     [-maxDynamicPower value]
     [-lpPowerAnalysisEffort]
     [-insertClockGating]
     [-clockGatingPrefix string]
```

```
[-decloneClockGating]
[-insertObsForClockGating]
[-maxGgForObs value]
[-noPle]
[-noReportQor]
[-multiMode | -noMultiMode]
[-readTcf fileName]
[-readTcfUpdate fileName]
[-optimizeYield]
[-noCheckBlackBox]
[-noCompressSDC]
[-postFixAssignOpt]
[-removeLoopBreakerCell]
[-zwl]
[-dftSetup fileName]
[-physicalScanChainOrdering scanchainSetupFile [-force]]
[-insertCompression compressionSetupFile [-force]]
[-def fileName]
[-cpf fileName]
```

Provides pre-placement synthesis capabilities inside SoC Encounter by using Encounter™ RTL Compiler Ultra to re-map and re-optimize the gate-level netlist to improve timing and area.

Parameters

`-appendWlmFile` *`fileName`*

Specifies the name of the wireload model definition file to be appended to the first library.

`-autoPathAdjust`

Enables the path-adjust flow in RTL Compiler to calibrate the slack delta against the Encounter slack. You must use this parameter in conjunction with the `-backEndReport` parameter.

Default: Off

`-backEndReport` *`FE_slack_report_file`*

Specifies the Encounter slack report file (`*.slk`) for the path-adjust flow in RTL Compiler. You must use this parameter in conjunction with the `-autoPathAdjust` parameter.

`-bypassPrecheck` Skips the timing precheck process. If you are using CTE, use this option.

`-clockGatingPrefix` *`string`*

Specifies a prefix to be assigned to clock-gating components inserted into the design. This option setting only takes effect if `-insertClockGating` is set.

`-cpf` *`fileName`*
Specifies the name of the CPF file.

`-cwlm`
Generates a custom wireload model and applies it to the current synthesis run. Creates the following files in the output directory:

`ec.wl.flat` Contains the description of the wireload model generated by the `-cwlm` option.

`ec.wl.flat.sdc`
This file is the constraint file which applies the custom wireload model to the design.

The `-cwlm` option overwrites these files each time it is used. If you have a wireload model and a constraint file that you want to use multiple times, you should save the files to different file names and then use the `-cwlmLib` and `-cwlmSdc` options to specify them. See the examples.

`-cwlmCellLimit` *`value`*

Specifies the minimum number of instances in a particular module before generating a custom wire load model. The default value is 10,000, which means that a custom wire load model will not be generated for a design module that has less than 10,000 instances.

`-cwlmLib` *`fileName`*
Specifies the name of an existing custom wireload model file. If you specify this file, you must also specify the corresponding constraints file.

`-cwlmSdc` *`fileName`*
Specifies the name of an existing constraints file for the custom wireload model. If you specify this file, you must also specify the corresponding wireload model file.

`-decloneClockGating`
Merges clock-gating instances driven by the same inputs. See the *Command Reference for Encounter RTL Compiler Ultra* for more information.

`-def` *`fileName`*

Specifies the name of the required DEF file.

Note: The DEF file contains flip-flops that must be placed in order to perform physical scan chain reordering or DFT compression.

`-dftSetup` *`fileName`*

Specifies the DFT setup file information for the design. You must specify the DFT setup file when using the `-physicalScanChainOrdering` or `-insertCompression` parameters.

`-effort low | medium | high`

Specifies the level of effort for the optimization run.

`low`

Best choice when you want quick feedback with a minimal number of changes, such as when the netlist is already in good shape and near the final stage. Use this option when you do not want to fully re-synthesize the netlist.

No re-mapping is done; critical paths are restructured, area reclaim is done, and incremental optimization is done to minimize changes to the netlist.

`medium`

Provides the best balance of quality of results versus runtime. Use this option when the netlist is reasonable but is not ready for timing closure and tape-out, or long logics on critical paths.

This effort level re-maps and restructures critical paths only, followed by incremental optimization. It also changes more of the global netlist and reduces area.

`high` (The default) Performs the most transformations in the netlist. Use this option when the netlist contains long logic chains, does not meet target timing and area objectives, or if you want to improve the timing and area from the current target.

This option completely re-maps and restructures the global netlist and then performs incremental optimization.

Expect the longest runtimes and a bigger area reduction if the original netlist synthesized poorly.

`-frontEndReport` *`RC_endpoint_report_file`*

Specifies the optional RTL Compiler endpoint report file that reports the slack of all timing endpoints in the design for the path-adjust flow. You can generate this file using the following RTL Compiler command:

```
report timing -endpoints -slack_limit 0
RC_endpoint_report_file
```

Note: The `-incrFirst` parameter of the `runVStorm` command automatically generates this file when launching RTL Compiler.

`-group` Groups the endpoints into bins based on their minimum and maximum slack values for the path-adjust flow. There are 10 bins for the constrained endpoints (minimum) and 10 bins for the relaxed endpoints (maximum). The endpoints in the design are constrained or relaxed based on the average of the minimum and maximum slack values specified in the bins. This parameter keeps the number of path-adjust exceptions to a constant value of 20 (less accurate path-adjust values). You must use this parameter in conjunction with the `-autoPathAdjust` parameter.

Default: On

`-incFirst {low | medium | high}`

	Performs an incremental optimization of the design before performing a full optimization of the design in the netlist-to-netlist flow. **Note:** In the netlist-to-netlist flow, RTL Compiler typically performs a full optimization followed by an incremental optimization of the design. Use this parameter if you have a need to run incremental optimization before running a full optimization in the netlist-to-netlist flow. Specifying a higher effort level increases the accuracy during incremental optimization, but also increases the run time. **Note:** This parameter automatically generates the *RC_endpoint_report_file* if an initial RTL Compiler synthesis run was not performed. For third-party netlists, you must perform incremental synthesis before generating the *RC_endpoint_report_file* with this parameter. *Default*: `medium`
`-inDir` *dirName*	Specifies the name of the input directory for all input files to RTL Compiler. The default is `./n2n_input`.
`-inheritCwlm`	When used with `-cwlm`, this option allows small, sub-modules to inherit the custom wireload model from the parent. This option is `off` by default.

`-insertClockGating`

Inserts clock-gating logic in a mapped netlist. See the Command Reference for Encounter RTL Compiler Ultra for more information.

`-insertCompression` *`compressionSetupFile`* `[-force]`

Specifies the compression setup file to perform DFT compression. You must use this parameter in conjunction with the `-dftSetup` parameter.

If you specify the *`compressionSetupFile`*, the Encounter software calls RTL Compiler and executes the following commands:

```
include compressionSetupFile
reload_cpf
execute_cpf
report dft_chains > dft/dftChainsCompression.rc # under /n2n_output/dft
write_scandef -version 5.5 > dft/dftCompressionScan.def # under /n2n_output/dft
```

If you do not specify the *`compressionSetupFile`*, the Encounter software generates an error message and will not perform DFT compression.

`-insertObsForClockGating`

Inserts and connects circuitry to improve the observability of the design after clock-gating logic is inserted. See the *Command Reference for Encounter RTL Compiler Ultra* for more information.

`-insertTieHiLo` Inserts tie high and tie low cells before writing out the netlist or sdc files.

`-lpPowerAnalysisEffort [medium | high | low]`

Controls the propagation of switching activities in the design.

`high`	Propagates switching activities with higher accuracy.
`medium`	The default. Propagates switching activities.
`low`	Does not propagate switching activities, and uses default settings instead.

`-maxCgForObs` *`value`* Specifies the maximum number of clock-gating cells that can be observed per observability flip-flop. The default value is 36. the *`value`* must be specified as a positive integer. To set this value, `-insertObsForClockGating` must also be set or an error condition is reported. See the *Command Reference for Encounter RTL Compiler Ultra* for more information.

`-maxDynamicPower` *`value`*

Turns on dynamic power optimization and specifies the maximum dynamic-power constraint of the design. Use a floating point number to set the value.

`-maxLeakagePower` *`value`*

Turns on leakage power optimization and specifies the maximum leakage-power constraint for the design. Use a floating point number to specify the value.

Setting a value for this option automatically triggers leakage power optimization. By default, `-maxLeakagePower` is not set.

Setting `-maxLeakagePower` in conjunction with `-multiVtEffort` triggers leakage power optimization with multi-vt. When `-multiVtEffort` is set to either low, medium, or high, unless you specify a value for `maxLeakagePower`, it is automatically set to 0.

`-mode auto|custom` Specifies either automatic or custom mode for running the command.

`auto`	Allows the interface to generate the RTL Compiler run scripts and setup directories, input/output files. This is the default.
`custom`	Users supply their own run script through the –`myScriptFile` option.

`-multiMode | -noMultiMode`

These options enable or disable multimode. When multimode is enabled, Encounter extracts multiple mode settings and files and passes them to RC. When multimode is disabled (default), the flow assumes a single-mode condition whether or not the current design in Encounter is set up for multimode.

Default: `-noMultiMode`

`-multiVtEffort [medium | high | low]`

Specifies the use of multiple-vt voltage libraries during leakage power optimization. To turn on multiple-vt leakage optimization, set either low, medium, or high. See the *Attribute Reference for Encounter RTL Compiler Ultra* for more information.

`high`	Performs a high-effort leakage power optimization and uses mainly high Vth cells, which have low leakage power and slow timing performance.
`medium`	Performs a medium-effort leakage power optimization using a combination of high and low threshold voltage cells.
`low`	Performs low-effort leakage power optimization using high Vth cells on cells on non-timing-critical paths and low Vth cells on timing-critical paths.

`-myScriptFile` *`fileName`*

Specifies a custom script instead of the automatically generated one. Must be used with the –`mode custom` option.

`-noCheckBlackBox`

Disables the software from checking the netlist for blackboxes.

`-noCompressSDC` Reads the Synopsys Design Constraint (SDC) file with advanced compression disabled and passes this file to RTL Compiler to create a cost group for each clock defined in the file.

`-noGroup` Specifies that the endpoints will not be grouped into bins for the path-adjust flow. Each endpoint is constrained or relaxed with a separate path-adjust exception. You must use this parameter in conjunction with the `-autoPathAdjust` parameter.

Default: Off

`-noInheritCwlm` When used with `-cwlm`, this option toggles off the `-inheritCwlm` option. This option is on by default.

`-noPle`	Performs a physical layout estimation for the wires based on a wireload model from `.lib` or a custom wireload model. *Default:* Performs a physical layout estimation for the wires based on information from the LEF file and the capacitance table.
`-noReportQor`	Specifies that ‘report qor’ command should not be appended to ec.tcl file.
`-noWriteSDC`	Disables generation of SDC file written from RC. The file is used for debugging purposes only.
`-opCond` *`name`*	Overwrites the default operating condition. If no default is defined, this option needs to be specified.
`-optFanout`	Runs fanout optimization before generating custom wireload models. Fanout optimization provides more realistic wireload data but will impact runtime. This option must be used with `-cwlm`.

`-optimizeYield`

Turns on yield optimization.

`-outDir` *`dirName`* Specifies the name of the directory for output files from RTL Compiler. The default is `./n2n_output.`

`-percentageEndPoints` *`percentage_number`*

Specifies the percentage of endpoints from the *`RC_endpoint_report_file`* (`-frontEndReport`) and the *`FE_slack_report_file`* (`-backEndReport`) that are to be relaxed or constrained in the path-adjust flow. You must use this parameter in conjunction with the `-autoPathAdjust` parameter.

Default: `20` percent

Note: The default selects 20 percent of the worst violating endpoints from the *`FE_slack_report_file`* and 20 percent of the non-critical endpoints from the *`FE_slack_report_file`* that are also violations in the *`RC_endpoint_report_file`*.

`-physicalScanChainOrdering` *`scanchainSetupFile`* `[-force]`

Specifies the scan chain setup file to perform physical scan chain reordering. You must use this parameter in conjunction with the `-dftSetup` parameter. The `-force` parameter forces the software to run physical scan chain reordering, even if the placement density is over 95 percent.

Note: The Encounter software automatically checks the placement density and will generate an error if the placement density is above 95 percent.

If you specifiy the *`scanchainSetupFile`*, the Encounter software calls RTL Compiler and executes the following commands:

```
include scanchainSetupFile
reload_cpf
execute_cpf
report dft_chains > dft/dftChains.rc # under /n2n_output/dft
write_scandef -version 5.5 > dft/dftChainsScan.def # under /n2n_output/dft
report dft_setup > dft/dftSetup # under /n2n_output/dft
write_atpg -cadence > dft/atpg # under /n2n_output/dft
write_dft_abstract_model > dft/abstractmodel # under /n2n_output/dft
```

If you do not specify the *`scanchainSetupFile`*, the Encounter software calls RTL Compiler and executes the following commands:

```
connect_scan_chains -physical
reload_cpf
execute_cpf
report dft_chains > dft/dftChains.rc # under /n2n_output/dft
write_scandef -version 5.5 > dft/dftChainsScan.def # under /n2n_output/dft
report dft_setup > dft/dftSetup # under /n2n_output/dft
write_atpg -cadence > dft/atpg # under /n2n_output/dft
write_dft_abstract_model > dft/abstractmodel # under /n2n_output/dft
```

`-postcheck`

Checks the zero load timing correlation between RTL Compiler and Encounter after optimization.

`-postFixAssignOpt`

Incrementally optimizes mapped gates to fix any high-fanout nets that were created by the `remove_assigns` command.

In the RTL Compiler flow, the `remove_assigns` command replaces assign statements in the gate-level netlist with buffers, which may cause large delays depending on the number of high-fanout nets created. Because there is no optimization step following the `remove_assigns` command in the RTL Compiler flow, violations may appear in the final timing report generated by the `runN2NOpt` command. This parameter fixes the high-fanout nets that were created by the `remove_assigns` command.

`-postloadInclude` *`fileNames`*

Specifies any one or more files to be included after loading data, including libraries, netlist, and constraint files.

`-postPlaceOpt` Specifies that post-placement optimization will be done in RC.

`-preloadInclude` *`fileNames`*

Specifies any one or more files to be included prior to loading data, including libraries, netlist, and constraint files.

`-preserveClockNetsAll`

Preserves all clock networks automatically.

`-preserveClockNetsNoData`

Preserves all clock networks automatically except for the paths arriving at data pin of flops.

`-preserveFixedInst` Places a preserve attribute on FIXED and COVER components.

`-preserveHierPinsWithSDC`

Preserves boundary opto to ports which have exceptions defined from SDC.

`-preserveSizingDifferentLibpins`

Supports libraries that contain functionally equivalent library cells with different pin names. When selected, an `ec.tcl` script automatically sources the proc `ec::set_dont_touch_on_instance_with_exp` to support libraries that contain functionally equivalent library cells with different pin names.

`-preserveTristateNets`

Preserves tristate nets.

`-rcAttr {`*`name_value_pairs`*`}`

Specifies any one or more name-value pairs for RTL Compiler internal attributes.

`-rcVar {`*`name_value_pairs`*`}`

Specifies any one or more name-value pairs for RTL Compiler internal variables.

`-readTcf` *`fileName`* Reads the toggle counts of the pins and nets in the specified TCF file and stores the assertions as net attributes for power estimation and optimization with RTL Compiler.

`-readTcfUpdate` *`fileName`*

Updates the probability values and toggle counts of the pins and nets in the specified TCF file.

`-removeLoopBreakerCell`

Removes loop breaker cells that were inserted by RTL Compiler.

`-removeTempFiles` Removes all the transferring files, including the script and log files.

`-report {all | none | timing | area | gate | design | zero | timemem | timingdetail | exception}`

Reports specified measurements along the optimization run. You can specify one or more report types.

`all`	All measurements
`area`	Area
`design`	Design
`exception`	Exceptions.
`gate`	Gate count.
`none`	No measurement.
`timemem`	Cpu-time and memory.
`timing`	Timing.
`timingdetail`	Detail timing end points and clock groups.
`zero`	Final zero-load timing.

`-saveToDesignName` *`name`*

Specifies the FE database name in which to store FE input and output files. The default is *`top_module`*`_n2n.enc.`

`-tieHiCell` *`cellName`*

Specifies the name of the tie high cell.

`-tieLoCell` *`cellName`*

Specifies the name of the tie low cell.

`-zwl` Ignores the wire load during netlist-to-netlist optimization. When selected, the wire load is set to zero.

Examples

The following example shows the recommended use of the `runN2NOpt` command.

```
runN2NOpt \
  -effort high \
  -inDir n2n.input \
  -outDir n2n.output \
  -report all \
  -saveToDesignName n2n.enc
```

The following command automatically generates a custom wireload model description file and a constraints file for the current synthesis run. The -optFanout switch performs timing-driven placement and runs fanout optimization to provide more realistic wireload data.

```
runN2NOpt -optFanout -cwlm
```

The following example shows how to manually generate the custom wireload model and then use it with `runN2NOpt` command:

```
loadConfig init_design.conf
placeDesign
optDesign -preCTS
wireload -file mycwlm -percent 1.0 -cellLimit 10000
freedesign
```

Then, to use the custom wire load model with N2N, use the following commands:

```
loadConfig init_design.conf
runN2NOpt -cwlm -cwlmLib mycwlm.flat -cwlmSdc myclwm.flat.sdc
```

The following command performs a high-effort multi-vt optimization with maximum leakage power constrained at 70 mW, maximum dynamic power constrained at 80 mW, and analysis effort set for high accuracy.

```
runN2NOpt -bypassPrecheck
-multiVtEffort high
-maxLeakagePower 70000000
-maxDynamicPower 80000000
-lpPowerAnalysisEffort high
```

19

Partition Commands

addNetToNetGroup

```
addNetToNetGroup
     netGroupName
     -net {netName | netNameList}
```

Adds a net to specified net group that was created with the createNetGroup command. A net can be a single net, bus name, or segment of a bus.

Note: If a net is a part of a net group for one partition, the net cannot belong to a net group for any other partition, even if the net connects pins of multiple partitions.

Parameters

netGroupName	Specifies the name of the created net group.
-net {*netName* | *netNameList*}	Specifies the name of an individual net, bus, or segment of a bus. You can use wildcards (*?) with this parameter.

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - Setting Pin Constraints
  - Net Group

Examples

- The following command adds net ENOUT to net group bundle1:

  ```
  addNetToNetGroup bundle1 -net ENOUT
  ```

- The following command adds the entire bus cpua to net group bundle2:

  ```
  addNetToNetGroup bundle2 -net cpua
  ```

- The following command adds only bus segments 2 through 10 to net group bundle1:

  ```
  addNetToNetGroup bundle1 -net cpua[2:10]
  ```

addPinToPinGroup

```
addPinToPinGroup
     [-cell cellName]
     -pinGroup pinGroupName
     -pin {pinName | pinNameList}
```

Adds a module port to specified pin group. You can use this command for the top-level modules also.

Parameters

`-cell` *`cellName`*
: Specifies the name of a module. This parameter is not required only if the module is the top-level module.

`-pinGroup` *`pinGroupName`*
: Specifies the name of pin group for the module.

`-pin` *`pinName`* | *`pinNameList`*
: Specifies the name of a module port. You can specify wildcards (*) with this parameter.

Example

The following command adds the module `SH17`'s port `RASN` to pin group `pin17`:

```
addPinToPinGroup -cell SH17 -pinGroup pin17 -pin RASN
```

alignPtnClone

```
alignPtnClone
     ptnName
     [-snap]
```

Aligns partition clones with the master partition on a power mesh. You can use this command after specifying a master partition and clone partition(s).

Note: The power mesh must be regular to for `alignPtnClone` to work correctly.

Parameters

ptnName	Specifies the name of the partition clone.
-snap	Snaps the partition clone to the nearest power grid. The Encounter software extracts the power grid and the snapping grid as specified in the Preferences form. **Note:** This does not necessarily snap partition clones to current snapping grid. The clones have same horizontal and vertical distance from the snapping grid as the master partition; therefore, if the master is not located at the snapping grid, neither will the partition clones However, you can snap all partition clones by using the Snap Floorplan form. This aligns master partitions and clone partitions to the power grid.

assembleDesign

```
assembleDesign
     {{-topDir topDesignDir}
          {{-blockDir blockDesignDir}... | {-blockData defName netlistName}...}
          [{-topFP | -chipFP} FPFile]
          [-fe]} |
     {-topDesign oaTopLib oaTopCell oaTopView
          {-block oaBlockLib oaBlockCell oaBlockView}...}}
     [-row]
     [-saveEcoRef [-ecoRefDir ecoDir]]
```

Brings back specified block data to the top-level design for chip assembly. You should run this command from the directory with the full chip-level floorplan for the top-down hierarchical flow.

You can run this command for designs that use the FE database as well as for designs that use the OpenAccess database.

This command can bring back row information from the block-level design with the `-row` parameter.

Note: The `assembleDesign` command does not support mixed-origin design. That is, all partitions and top should have either a center origin or a lower-left origin.

You can use this command after implementing blocks and top-level designs.

Parameters

`-block` *`oaBlockLib oaBlockCell oaBlockView`*

Specifies the block-level library, cell, and view names respectively for a design that was saved into an OpenAccess database.

This parameter is used when bringing back block data for the partition from an OpenAccess database.

Note: If you specify more than one block-level design, specify the `-block` option separately for each block. For example:

```
-block libForOA TDSP_CORE ptnView1 -block libForOA
TDSP_ARB ptnView1
```

`-blockData` *`defName netlistName`*

Specifies the DEF file name and the Verilog netlist file name. Use this parameter if the Verilog netlist file name or the DEF file name is different from the cell or module name. You must provide the full path (absolute or relative) with the file names.

Note: If you specify more than one block-level design, specify the `-blockData` option separately for each block-level DEF file and Verilog netlist file. For example:

```
-blockData defFile_1 VerilogFile_1 -blockData
defFile_2 VerilogFile_2
```

`-blockDir` *`blockDesignDir`*

Specifies the path to the block-level design(s). This directory should be generated by the `saveDesign -def` command.

Note: If you specify more than one block-level design, specify the `-blockDir` option separately for each block-level directory. For example:

```
-blockDir ptnDir1 -blockDir ptnDir2
```

`-fe`

Specifies that Encounter place-and-route data should be used.

Note: The `-fe` parameter cannot be used with the `-blockData` parameter because the `-fe` parameter points to data from the Encounter database format whereas the `-blockData` parameter points to data from Verilog netlist and DEF name.

Default: By default, data from DEF files is used.

`-row`

Specifies that row information should be brought back from the block-level design.

`saveEcoRef [-ecoRefDir` *`ecoDir`*`]`

Generates a partition DEF file for each partition in the user-specified ECO reference directory.

This parameter is used for the Active Logic View flow. For more information, see the Steps to Run ART-based Post-Route Optimization section in the "Using Active Logic Views in Hierarchical Designs" chapter of the *Encounter User Guide*.

If both the `-saveEcoRef` and the `-ecoRefDir` are specified, the `assembleDesign` command creates a directory as specified with the `-ecoRefDir` parameter and creates a file name *`ptnName`*`_eco_ref.def`. in this directory (*`ptnName`* is the name of the partition or module).

If only the `-saveEcoRef` parameter is specified, the `assembleDesign` command creates the file named *`ptnName`*`_eco_ref.def` in the current working directory.

`-topDesign` *`oaTopLib oaTopCell oaTopView`*

Specifies the top-level library, cell, and view names respectively for a design that was saved into an OpenAccess database.

This parameter is used when bringing back top-level data from an OpenAccess database.

`{-topFP | -chipFP}` *`fpFile`*

	Provides the chip-level floorplan file. By default, the command reads in the floorplan file from the provided top-level design. You should use this option for the top-down hierarchical flow. **Note:** The `-topFP` and the `-chipFP` parameters perform the same function. The `-topFP` parameter has been retained for backward compatibility. **Note:** From the 8.1 USR1 release onwards, the `savePartition` command saves the partition definitions in the chip-level and the block-level floorplans. Therefore, you do not need to specify the chip-level floorplan while assembling the design for the top-down flow. However, the `-topFP` and the `-chipFP` parameters are still supported for the files saved with `savePartition` command in the earlier releases.
`-topDir` *`topDesignDir`*	Specifies the path to the top-level design created by `saveDesign -def`. This design directory should contain the Verilog netlist and DEF files.

Related Topics

- Partitioning the Design chapter in the *Encounter User Guide*
  - Flow Methodologies
  - Handling of Blackboxes with Non-R0 Orientation
  - Working with OpenAccess Database
- Assemble the Design and Sign Off in the *Encounter Hierarchical Implementation Flow Guide*

Examples

- The following command assembles the design after bringing back information from the top-level cell and two block-level cells. The top-level design data is in the `dirTop` directory and the block-level data is in the `dirBlock1` and `dirBlock2` directories.

  ```
  assembleDesign -topDir dirTop -blockDir dirBlock1 -blockDir dirBlock2
  ```

- The following command assembles a design with the following block-level information:
  - the DEF file with the name `def_file`

- the verilog netlist `v_netlist.v`

```
assembleDesign -topDir dirTop -blockData def_file v_netlist.v
```

- The following command generates a partition DEF file for each partition in the ECO_REFERENCE_1 directory.

```
assembleDesign -topDir dirTop -blockDir dirBlock1 dirBlock2 -saveEcoRef
-ecoRefDir ECO_REFERENCE_1
```

- The following command assembles a design that was saved in an OpenAccess database. The design contains two blocks.
  - The library, cell, and view names for the top-level are `libForOA`, `DTMF`, and `ptnView1` respectively.
  - The library, cell, and view names for the first block are `libForOA`, `TDSP_CORE`, and `ptnView1` respectively.
  - The library, cell, and view names for the second block are `libForOA`, `TDSP_ARB`, and `ptnView1` respectively.

```
assembleDesign -topDesign libForOA DTMF ptnView1 -block libForOA TDSP_CORE
ptnView1 -block libForOA TDSP_ARB ptnView1
```

assignIoPins

```
assignIoPins
     [-noPinsBelowStrip]
     [-pinsOffStrip]
     [-minRouteLayer layerId]
     [-align]
     [-maxRouteLayer layerId]
     [-pin pinList [-moveFixedPin]]
```

Legalizes the top I/O pin locations for a block-level design. If you set the maximum or minimum routing layer while running Trial Route or Nano Route, the command automatically uses these constraints.

By default, the `assignIoPins` command sets the depth of all partition pins equal to: `MINAREA/pin width`. This ensures that the minimum area rule is met. However, if any pin-size-related constraints are specified on these pins, the user-specified pin size will have a higher precedence.

You can use this command after running placement and trial routing to legalize the I/O pins.

Parameters

`-align`

If you specify this option, each top I/O pin will be aligned with one of the instance pins with which it is connected. The instance pin with which the I/O pin is aligned is the one whose distance from its nearest edge is shortest among all the instance pins connected to the top I/O pin.

Default: If you do not specify this option, the location of the top pin is based on the average of the distance of all instance pins (that are connected to the top I/O pin) from their respective nearest edges.

`-maxRouteLayer layerId`

Specifies the highest layer on which I/O pins can be assigned.

For example, if the value of `layerId` is 5, the highest layer on which I/O pins are assigned is layer 5.

Note: The `maxRoute` layer parameter takes a higher precedence over any maximum layer constraint specified through `setTrialRouteMode -maxRouteLayer` or through `setNanoRouteMode -routeTopRoutingLayer`.

`-minRouteLayer layerId`

	Specifies the lowest layer number on which I/O pins can be assigned. For example, if the value of *`layerId`* is 4, the lowest layer on which I/O pins are assigned is layer 4. **Note:** The value of *`layerId`* for the `-minRouteLayer` parameter cannot be less than 2. **Note:** The `minRoute` layer parameter takes a higher precedence over any minimum layer constraint specified through `setTrialRouteMode -minRouteLayer` or through `setNanoRouteMode -routeBottomRoutingLayer`.
`-moveFixedPin`	If you use this parameter, the specified pins with a *Fixed* status can also be moved. **Note:** Pins that are not specified but have a *Fixed* status are *not* moved. *Default*: If you do not specify this parameter, the specified pins with a *Fixed* status are *not* moved.
`-noPinsBelowStrip`	Prevents the generation of pins below power and ground strips.
`-pin` *`pinList`*	Specifies that only the specified pins should be assigned. *Default*: If you do not specify this parameter, all pins are assigned.
`-pinsOffStrip`	Prevents the generation of pins on all metal layers, above and below the power and ground strips.

Related Topics

- Partitioning the Design chapter in the *Encounter User Guide*
  - Block Implementation
  - Assigning I/O Pins

Example

The following command legalizes the top IO pin locations and specifies that the lowest layer on which IO pins can be assigned is layer 2:

```
assignIoPins -moveFixedPin -minRouteLayer 2
```

assignPtnPin

```
assignPtnPin
     [-pinsOffStrip]
     [-noPinsBelowStrip]
     [-skipPinRefine]
     [-markFixed]
     [-enforceRoute]
     [-noPinLayerOverlap]
     [-printPinMovementStatistics]
     [-basedOnMasterOnly]
     [-maxPinMovementForAlign]
     [-improvePinOrder]
     [partitionName ... | {-ptn partitionName -pin pinList}... [-moveFixedPin]]
```

Assigns partition pins before the partitions are committed. By default, the assigned pins will be in a *Placed* status after running this command. You can specify a *Fixed* status by running this command with the `-markFixed` option.

Note: This command assign pins for blackboxes as well as regular partitions.

Running the `saveFPlan` command saves the pin assignment in the floorplan to a separate file named *`fpFileName`*`.fp.ptnpin`, which is included in main *`fpFileName`*`.fp` floorplan file.

By default, the `assignPtnPin` command sets the depth of all partition pins equal to: `MINAREA/pin width`. This ensures that the minimum area rule is met. However, if any pin-size-related constraints are specified on these pins, the user-specified pin size will have a higher precedence.

You can use this command after specifying the partitions, placing the design, and after running or loaded routing information. If no routing information is available, the Encounter software uses the projected pin assignment

Note: This command only assigns signal pins. Power and ground pins are created during partition. However, this pin optimization honors power stripes and followpins.

Note: If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.

Note: The `assignPtnPin` command internally calls the `reportUnalignedNets` command to generate the summary report at the command line.

Parameters

`-basedOnMasterOnly` Specifies that for designs that have partition clones, the command should consider only the partition master and use the master assignment for all the clones.

Default: For designs that have partition clones, this command by default also considers partition clones while assigning pins.

Note: For blackbox pin assignment, the command considers only the partition master by default.

`-enforceRoute` Specifies that the pin assignment completely follows the routing results generated by Trial Route, without taking into consideration any pin constraints.

For example, the following constraints are not considered when the `-enforceRoute` parameter is used:

- Constraints related to pin spacing, layer, width, depth, and side
- Pin Guide constraints and Pin Group constraints
- Constraints specified by the `-pinsOffStrip` parameter of the `assignIoPins` and the `assignPtnPin` commands

The `-enforceRoute` parameter can be useful when, for example, you want to compare pin locations based purely on Trial Route results with pin locations that honor the specified pin constraints.

`-improvePinOrder` Reduces flight line crossing during pin alignment.

The partition and the blackbox pins are rearranged to be aligned with external connections:

- Guided pins are reordered within the pin guide box.
- Pins are rearranged at (or close to) their original locations.
- Pin ordering is performed independently for each partition edge.

The locations of fixed pins, aligned pins or multi-location pins is not changed. Master-clone pins are not reordered.

`-markFixed` Specifies that all pins will have a *Fixed* status.

`-maxPinMovementForAlign`

Specifies the maximum distance, in microns, by which the pins can be moved away from the routing cross point during pin assignment.

Note: This parameter applies only to route-based pin assignment.

This parameter can be useful if you want to limit the distance by which pins could be away from routing cross points in order to ensure pin alignment.

`-noPinsBelowStrip` Prevents the generation of pins below power and ground strips.

`-noPinLayerOverlap` Specifies the following:

- There are no pins above or below any pin, across layers. That is, if a pin on a layer has co-ordinates x and y, there are no pins on co-ordinates x and y on any other layer. This applies to all pins on all layers.
- If the value of the pin pitch is 1 (instead of the default value of 2 or any other value), adjacent pins are *not* placed on the same layer.

`partitionName ...` Specifies the partition names for pin assignment.

Default: Pins are assigned for all partitions.

`-ptn` *`partitionName`* `-pin` *`pinList ...`*

Specifies the names of the partitions and the list of pins to be assigned in the respective partitions.

Default: Pins are assigned for all partitions.

`-moveFixedPin` If you use this parameter, the specified pins with a *Fixed* status can also be moved.

Note: Pins that are not specified but have a *Fixed* status are *not* moved.

Default: The specified pins with a *Fixed* status are *not* moved.

`-pinsOffStrip` Prevents the generation of pins on all metal layers, above and below the power and ground strips.

`-printPinMovementStatistics`

Prints a summary of the deviation (in micrometers) of the final pin locations from the point of intersection of the net and the partition edge.

The report on the screen contains the following statistics:

- Maximum displacement on the X axis. This is the difference in the x-coordinates of the point of intersection of the net and the partition edge, and the point at which the pin is finally assigned.
- Maximum displacement on the Y axis.
- Maximum X+Y displacement. The X+Y displacement of a pin is the sum of the displacement of a pin on the X axis and the displacement of the pin in the Y axis. For example, if a pin has moved 10 micrometers on the X axis and 5 micrometers on the Y axis, the X+Y displacement for that pin is 15. Thus, if there are three pins and their X+Y displacements are 5, 23, and 30 respectively, the maximum X+Y displacement is reported as 30.
- Mean of the X+Y displacement.

The report appended to the Encounter log file contains the following additional details:

- Names of the pins that have moved the maximum distances.
- The coordinates of the point of intersection of the net and the partition edge, and the point at which the pin is finally assigned.

Here is a sample report displayed on screen.

```
Partition: [tdsp_core]
maximum X   =   120.1200 um (Pin: t_data_out[6])
maximum Y   =   58.8000 um (Pin: rom_data_in[4])
maximum X+Y =   154.5600 um (Pin: t_data_out[6])
moved from (706.2000, 1025.0800) -> (586.0800, 990.6400)
mean X+Y    =   12.5600 um
total pins  =   114
pins moved  =   75
```

The log file has additional information such as:

```
Pin: int: X Movement: 3.3000 um ; Y Movement: 0.0000. From
(701.2500, 1247.6800) -> (697.9500, 1247.6800)
```

`-skipPinRefine`	Does not allow topology-based pin refinement. Pin refinement refines the pin locations to reduce the top-level congestion or top-level wire length. Pin location assignment is only based on connectivity or trial routing.

Related Topics

- Partitioning the Design chapter in the *Encounter User Guide*
  - Chip Planning
  - Blackbox Flow
  - Handling of Blackboxes with Non-R0 Orientation
  - Route-based Pin Assignment
  - Tips for Assigning Partition Pins
  - ECO Pin Assignment

blockPtnSidePinLayer

```
blockPtnSidePinLayer
     ptnName
     {n | w | s | e}
     layerId
```

Blocks the generation of pins on a metal layer on one side of a partition. This command is used in conjunction with `createPinGuide` which generates pins on the desired metal layer for a partition. You can use this command during partition floorplanning.

Parameters

layerId	Specifies the metal layer to be blocked.
n | w | s | e	Specifies the side of the partition. n specifies the side as North side, w as West side, s as South side, and e as East side. *Default:* If you do not specify this parameter, all sides are selected.
ptnName	Specifies the name of the partition.

Example

The following command blocks pins to be created on the North side of partition `sheet17`:

```
blockPtnSidePinLayer sheet17 n 2
```

changeBBoxMasterToR0

```
changeBBoxMasterToR0
     [-checkOnly]
     [cellName | {cellNameList}]
```

Converts the orientation of the master instance blackboxes to R0.

Partitioning-related commands such as assignPtnPin, partition, assembleDesign, flattenPartition, convertBlackBoxToFence, and editPin work only with those blackboxes whose master instances have an R0 orientation. If the design has master blackboxes whose orientation is non-R0, you can use the changeBBoxMasterToR0 command to change the orientation of these master blackboxes to R0.

Note: Orientations of corresponding clone instances are also transformed to R0.

Parameters

`-checkOnly`

Specifies that the command will not actually convert the orientation of any master blackbox; it only displays the number of master blackboxes whose orientation would have been changed had the command been run without the `-checkOnly` parameter.

Default: The orientation of master blackboxes in the design are converted to R0.

`cellName`|{*`cellNameList`*}

Specifies that cell, or the list of blackbox cells, whose orientation will be converted to R0.

Default: The orientation of all master blackboxes is converted to R0.

Related Topics

- Partitioning the Design chapter in the *Encounter User Guide*
  - Handling of Blackboxes with Non-R0 Orientation

Examples

- The following commands converts the orientation of all master blackboxes that have a non-R0 orientation to R0.

```
changeBBoxMasterToR0
```

- The following commands converts the orientation of the master blackboxes CORE_CELL120 and CORE_CELL121 to R0.

  ```
  changeBBoxMasterToR0 {CORE_CELL120 CORE_CELL121}
  ```

- The following command uses the `-checkOnly` parameter. The command, therefore, does not actually convert the orientation of any master blackboxes; it only returns the number of blackboxes whose orientation would have been converted to R0 had the command been run without the `-checkOnly` parameter.

  ```
  changeBBoxMasterToR0 -checkOnly
  ```

checkHierRoute

```
checkHierRoute
     [-ptn {ptnName1 ptnName2 ...}]
     [-outFile outputFileName]
```

Checks and reports hierarchy violations for nets in the specified partitions.

The following conditions for hierarchy violations are checked:

- Intra-partition nets: All routes of this net should be within the partition boundary.
- Pure-Top nets: All routes of this net should be outside the partition boundary or touch the partition boundary from *outside*.
- Inter-partition nets: At least one route of this net must cross the partition boundary at partition pin. No route should cross the partition boundary at any other point.

Important

You can use this command to check for hierarchy violations before you push down the signal routes with the `partition -pushRoute` command.

Parameters

`-ptn {ptnName1 ptnName2 ...}`

Report the nets that violate the hierarchy with respect to the specified partitions.

Default: The command checks the hierarchy violations for nets on all partitions.

`-outFile outputFileName`

Prints the details of net-wise violations in the specified file.

Default: By default, the name of the file is `hier.rpt`.

Related Topics

- Partitioning the Design chapter in the *Encounter User Guide*
  - Pushing Down Signal Routes

Example

The following command checks the violations for nets on all the partitions in the design. A report of the violations is printed in the `hier.rpt` file.

```
checkHierRoute
```

The output of this command on the display terminal is as follows:

```
------------------------------------------------------------
Summary of hierarchy violating nets with respect to each partition:
Partition: p1: 3 nets (0 intra, 2 inter, 1 top, 0 global)
Partition: p2: 1 nets (0 intra, 1 inter, 0 top, 0 global)
Partition: p3: 2 nets (1 intra, 0 inter, 1 top, 0 global)
Details of hierarchy violating nets reported in: hier.rpt
------------------------------------------------------------
```

The report printed in the `hier.rpt` file is as follows:

```
------------------------------------------------------------
Added the new header as follows:
#####################################################################
#  Generated by:      Cadence First Encounter 08.10-b060_1
#  OS:                Linux i686(Host ID icdlnx08s)
#  Generated on:      Mon Sep  8 09:41:49
#  Command:           checkHierRoute
#####################################################################
Partition: p1 . List of hierarchy violating nets against this partition:
Hierarchy violating net   Category   Hierarchy violating wire box(llx lly urx ury)
-----------------------   -----   ----------------------------------------
inter_ok_n1                 Inter-Partition   "No wire crossing partition pin"
inter_hierarchy_violating_n Inter-Partition   (2189.41 2549.82 2447.09 2550.10)
top_hierarchy_violating_n2  Top               (1192.15 2310.14 1192.43 2695.14)

Partition: p2 . List of hierarchy violating nets against this partition:
Hierarchy violating net   Category   Hierarchy violating wire box(llx lly urx ury)
-------------------------------------------------------------------------------
inter_ok_n1             Inter-Partition         "No wire crossing partition pin"
-------------------------------------------------------------------------------
```

checkPinAssignment

```
checkPinAssignment
     [-ptn ptnName]
     [-pin {pinName | pinNameList}]
     [-pinMinAreaCheck {0 | 1}]
     [-pinSpacingCheck {0 | 1}]
     [-pinLayerCheck {0 | 1}]
     [-pinWidthCheck {0 | 1}]
     [-pinDepthCheck {0 | 1}]
     [-pinGuideCheck {0 | 1}]
     [-pinOnFenceCheck {0 | 1}]
     [-pinOnTrackCheck {0 | 1}]
     [-verbose]
     [-outFile fileName]
```

Checks the generated partition pins and I/O pins for the following:

- Pins are assigned to the proper metal layer depending on the side of the partition
- Pins are assigned within the range of layers reserved for the partition
- Any particular location on a routing layer has only one pin assigned
- Pins are assigned on the routing track
- Pins have a valid spacing from:
  - routing blockages
  - power pins and ground pins
- Pins have the correct size, status, and type
- Pin assignment follows the minimum area rule
- If the `-minArea` parameter is specified, the command checks for and reports the pins that are:
  - floating from inside (that is, the pins that are connected only to the top level)
  - floating from both inside the partition and at the topl level.

The summary report generated by the command also includes information about feedthrough pins.

You can use this command after running trial routing and then partitioning with pin assignment.

Note: If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will check violations only with

respect to the first shape and will ignore all other shapes. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the command will check violations only with respect to the first LEF port and will ignore all other LEF ports.

Parameters

`-outFile` *`fileName`*
: Specifies that in addition to displaying the report on screen, the command should also print the output in the specified file.

`[-pin {`*`pinName | pinNameList`*`}]`
: Specifies that the command should be run on the specified pin(s) only.

`-pinDepthCheck {0 |1}`
: Specifies whether the command should check for violation of pin depth constraints.

  Default: 1 (The check is performed.)

`-pinGuideCheck {0 |1}`
: Specifies whether the command should check for violation of pin guide constraints.

  Default: 1 (The check is performed.)

`-pinLayerCheck {0 |1}`
: Specifies whether the command should check for violation of pin layer constraints.

  Default: 1 (The check is performed.)

`-pinMinAreaCheck {0 | 1}`
: Specifies that the command checks for and reports the pins that are:

  - floating from inside the partition
  - floating from both inside the partition and at the top level

  Default: 1 (The check is performed.)

`-pinOnFenceCheck {0 |1}`

Specifies whether the command should check whether pins are placed on fence.

Default: 1 (The check is performed.)

`-pinOnTrackCheck {0 |1}`

Specifies whether the command should check whether pins are placed on layer tracks.

Default: 1 (The check is performed.)

`-pinSpacingCheck {0 |1}`

Specifies whether the command should check for violation of pin spacing constraints.

Default: 1 (The check is performed.)

`-pinWidthCheck {0 |1}`

Specifies whether the command should check for violation of pin width constraints.

Default: 1 (The check is performed.)

`-ptn` *`ptnName`* Specifies the partition for which the pins will be checked. This parameter is required if the partition is not the top-level module.

`-verbose` Provides detailed pin-specific information for each reported violation.

Related Topics

- Partitioning the Design chapter in the *Encounter User Guide*
  - Assigning Pins
  - Assigning I/O Pins
  - Validating Pin Placement

clearActiveLogicView

`clearActiveLogicView`

Resets the reduced timing graph created by `createActiveLogicView` back to the original timing graph.

Parameters

None

clearVirtualPartition

`clearVirtualPartition`

Resets the reduced timing graph created by `createVirtualPartition` back to the original timing graph.

Parameters

None

Note: The `clearVirtualPartition` command is now obsolete and has been replaced by `clearActiveLogicView`. This obsolete command works in this release, but to avoid warnings and ensure compatibility with future releases, update your script to use `clearActiveLogicView.`

connectMacroFeedthrough

```
connectMacroFeedthrough
     [{-selectNet fileName} | {-excludeNet fileName} | -selectMarkedNet]
     [{-selectInst fileName} | {-excludeInst fileName}]
     [{-ecoFile fileName} | -checkOnly]
     [-portMap fileName]
     [-verbose]
     [-maxSearchDistance value]
     [-searchAllSides]
     [-abutment abutmentDistance]
     [-abutmentFile fileName]
     [-floatingPortList fileName]
```

Connects predefined feedthrough pins in custom macros to overlapping nets that have wires crossing over the custom macros, based on the routing information.

The feedthrough connectivity details are defined in a mapping file, which is specified through the `-portmap` *`filename`* parameter. You can include or exclude specific nets or instances.

This command deletes routing only for the modified nets.

You can run this command after placing the design and running Trial Route.

Parameters

`-abutment` *`abutmentDistance`*

Specifies the distance that pins on the same track and layer must have from one another so as to be considered as abutted. If two or more pins are considered as abutted, any feedthrough path that passes through one of the pins will pass through the other pin(s) as well.

Default: If this parameter is not specified, the default value is 1 micron.

`-abutmentFile` *`fileName`*

Reports, in the file specified, the list of pins that are considered as abutted. In the file, each feedthrough path is traced from the first input pin to the last input pin. Each path is preceded by the word `Begin` on a new line and followed by the word `End` on a new line.

Here is the extract from an example file:

```
Begin
Inst [Inst1], Term In [A1] Term Out [X1]
Inst [Inst2], Term In [A2] Term Out [X2]
End
```

In the previous example, the out term `X1` of instance `Inst1` is abutted with the in term `A2` of instance `Inst2`.

`-checkOnly`

Specifies that only a report of the messages should be displayed but no changes should be made to the netlist.

`-ecoFile` *`fileName`*

Specifies the name of the ECO file that will contain a report of the changes. You can use this file while performing ECO changes.

Here is a sample report generated through this parameter:

```
FORMATVERSION 2


ATTACHTERM eco_instance1 inputbuf2[2] testnet
ADDNET FE_FTM_1_testnet
ATTACHTERM eco_instance1 outputbuf2[2]
FE_FTM_1_testnet ATTACHTERM i1 A FE_FTM_1_testnet
ATTACHTERM eco_instance1 rstb FE_FTM_1_testnet
```

`-excludeInst` *`fileName`*

Prevents feedthrough buffers from being added for the specified instance. The names of the instances are specified in a file, specified by *`fileName`*.

In the file, you can specify wildcards (*) for the instance names to exclude.

Default: All instances of the cell types specified in the portmap file are considered.

`-excludeNet` *`fileName`*

Prevents feedthrough buffers from being added to the specified net. The names of the nets are specified in a file, specified by *fileName*.

In the file, you can specify wildcards (*) for the net names to exclude.

Default: All nets or bus names are considered.

`-floatingPortList` *fileName*

Specifies that the list of unused ports should be saved in the specified file. The file lists the instance name and the pin name for each unused port.

After running the `connectMacroFeedthrough` command, the unused ports should be manually tied. You can use this parameter to save the list of the unused ports to a file.

`-includeInst` *fileName*

Specifies that feedthrough buffers should be added only for the specified instances. The names of the instances are specified in a file, specified by *fileName*.

In the file, you can specify wildcards (*) for the instance names to include.

Default: All instances of the cell types specified in the portmap file are considered.

`-includeNet` *fileName*

Specifies that feedthrough buffers should be added only to the specified nets. The names of the nets are specified in a file, specified by *fileName*.

In the file, you can specify wildcards (*) for the net names to include.

Default: All nets or bus names are considered.

`-maxSearchDistance` *value*

Specifies the distance in microns from the wire crossing till which the command should search to find a connected feedthrough.

Note: Specifying a large value for this parameter might have an adverse impact on the Quality of Results (QoR).

`-portMap` *`fileName`*

Specifies the name of the file that contains the feedthrough connectivity definitions.

Default: If you do not specify a file name, a file with the name `portmap` in the current directory is used.by default.

Note: A file with the feethrough connectivity definitions must be available.

The syntax of the file is as follows:

```
MACRO MacroName
Macro definition section
END MACRO
```

The definition of the macro is provided in the *`Macro definition`* section, which can contain one or more feedthrough sections. The name of the feedthrough section is optional.

The syntax of the Feedthrough section is as follows:

```
Feedthrough [FeedthroughName]
Pin Section
END FEEDTHROUGH
```

Each Feedthrough section contains one section for the INPUT PIN and one section for the output pin.

Note: Multi-fanout feedthrough sections are not supported.

The syntax of the pin section is as follows:

```
PIN PinName
DIRECTION {in | out };
SIDE {left | right | top | bottom};
END PIN
```

If the pin direction is not specified, it is read from the LEF file. If you specify a value for pin direction, ensure that it is consistent with the description in the LEF file.

Note: The pin direction `inout` is not supported.

Note: All the predefined macro feedthrough pins should be floating pins.

Here is an example of a mapping file:

```
MACRO RAMXXX
FEEDTHROUGH feedthrough1
PIN feedthrough1_in
END PIN
PIN feedthrough1_out
END PIN
END FEEDTHROUGH
FEEDTHROUGH feedthrough2
PIN feedthrough2_in
END PIN
PIN feedthrough2_out
END PIN
END FEEDTHROUGH
END MACRO
```

`-searchAllSides`

Specifies that if the closest available feedthrough pin on the same side cannot be connected, the command should search for the feedthrough pins on other sides as well.

When you specify this parameter, it is recommended to limit the distance of the search using the -maxSearchDistance option to get good QoR.

Default: The command searches for a feedthrough whose in and out pins lie *on the same sides* of the macro on which the wires enter and exit the macro.

`-selectMarkedNet`

Specifies that feedthrough buffers should be added only to the nets that are currently selected.

Default: All nets.are considered.

`-verbose`

Provides additional messages, for example net-specific messages.

Example

The following command connects predefined feedthrough buffers in custom macros for nets that are listed in the file named `NetsToIncludeFile`. The command searches for a distance of 10 microns. The feedthrough connectivity is described in a file named `portmapFileForNets`.

```
connectMacroFeedthrough portMap portmapFileForNets -selectNet NetsToIncludeFile
-maxSearchDistance 10
```

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - "Utilizing Pre-Defined Feedthrough Buffers in Custom Macros"

convertBlackBoxToFence

```
convertBlackBoxToFence
     {-cell cellName | -inst instName}
```

Converts a blackbox to a fence. Once the black box netlist is well defined, use this command to convert the black box back to a regular partition.

Parameters

`-cell` *`cellName`*	Specifies the name of a module or submodule that should be converted to a fence.
`-inst` *`instName`*	Specifies the name of the blackbox instance that should be converted to a fence.

Example

```
convertBlackBoxToFence -inst hinst5
```

convertFenceToBlackBox

```
convertFenceToBlackBox
     {-cell cellName | -hinst hinstName}
     [-size { x y }]
```

Converts a fence to a blackbox. Use this command before committing partition.

Parameters

`-cell` *`cellName`*	Specifies the name of a module or submodule that should be converted to a black box
`-hinst` *`hinstName`*	Specifies the name of the hierarchical instance that should be converted to a black box
`-size` {*x y*}	Specifies the width and height of the blackbox. *Default:* If you do not specify this parameter, the current fence size is taken as the initial size of the blackbox.

Example

```
convertFenceToBlackBox -cell cp1A -size 12 44
```

createActiveLogicView

```
createActiveLogicView
     [help]
     -type flatTop
```

Trims the timing graph to ignore logic inside the first level registers in partitions to improve runtime for timing-related commands.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `createActiveLogicView` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man createActiveLogicView`.
`-type flatTop`	Marks the top-level timing graph to mask all logic inside the interface logic of each partition. `flatTop` is the only type you can specify. *Default:* If you do not specify this parameter, timing-related commands must operate on the entire flat netlist rather than on the virtual partition version.

Examples

For examples and more information, see “Chip-Level Budgets Derived by Using Active Logic View” in the Timing Budgeting chapter of the Encounter User Guide, and “Using Active Logic View for Chip-Level Interface Circuit Timing Closure” in the Timing Optimization chapter of the Encounter User Guide.

createILMDataDir

```
createILMDataDir
     -dir directory_name
     [-cell cell_name]
     [-setup | -hold | -setupHold | -mmmc]
     [-verilog cell_name.v]
     [-spef spef_file.spef]
     [-sdc sdc_file.sdc]
     [-viewName view_name]
     [-rcCornerName RC_corner_name]
     [-overwrite]
     [-incr]
     [-cts]
     [-si]
```

Creates an ILM directory from existing data.

If the data files already exist in the specified directory, the tool stops and generates an error message. You can overwrite existing files by specifying `-overwrite`. You can add files to the directory by specifying `-incr`.

You can specify that the software uses the data in the directory for setup, hold, MMMC, or both setup and hold analysis.

The command also creates a tcl script (`cell_name_ILM.tcl`) that you can source in your command file. The script contains the following command:

```
specifyIlm -cell cell_name -dir dir_name
```

Parameters

`-cell` *`cell_name`*	Specifies the name of the ILM module. If you do not specify a cell name, the software uses the `-verilog` *`cell_name`* for the module name.
`-cts`	Creates and stores data in the directory for clock tree synthesis (CTS).
`-dir` *`dir_name`*	Stores the ILM data in the specified directory: the absolute directory path or the path relative to the current directory.
`-hold`	Specifies that the data in the directory will be used for hold analysis. You cannot specify `-setupHold`, `-setup`, `-mmmc`, `-viewName`, or `-rcCornerName` when you use this parameter.

`-incr` Lets you add files in addition to the existing files to the directory.

`-mmmc` Specifies that the data will be used for MMMC analysis.

`-overwrite` Overwrites existing files in the directory.

`-rcCornerName` *`RC_corner_name`*

Specifies the RC corner name for MMMC analysis. You cannot specify `-setup`, `-hold`, or `-setupHold` when you use this parameter. The `-spef` parameter identifies the spef file to be used with the specified corner.

`-sdc` *`sdc_name.`*`sdc`

Specifies the name of the `.sdc` file for the analysis view.

`-setup` Specifies that the data in the directory will be used for setup analysis. You cannot specify `-hold`, `-setupHold`, `-mmmc`, `-viewName`, or `-rcCornerName` when you use this command.

`-setupHold` Specifies that the data in the directory will be used in both setup and hold analysis. You cannot specify `-hold`, `-setup`, `-mmmc`, `-viewName`, or `-rcCornerName` when you use this command.

`-si` Specifies that the data in directory will be used for signal integrity analysis.

`-spef` *`spef_name`*`.spef`

Specifies an existing `.spef` file that you want to include in the directory.

`-viewName` *`view_name`*

Specifies the MMMC view for the ILM module. You cannot specify `-setup`, `-hold`, or `-setupHold`. Specify the `-sdc` file for the analysis view.

`-verilog` *`netlist_name`*`.v`

Specifies the verilog file you want to include in the ILM data directory.

Examples

- The following commands create two module directories:

```
createILMDataDir -dir A -verilog V1.v -spef speffile1.spef -sdc sdcfile1
createILMDataDir -dir A -verilog V2.v -spef speffile2.spef -sdc sdcfile2
```

  The command creates two tcl files in the current directory.

The `module1_ILM.tcl` file contains the following command:

```
specifyIlm -dir A -cell module1
```

The `module2_ILM.tcl` file contains the following command:

```
specifyIlm -dir A -cell module2
```

- If you do not have Encounter database for an implemented block but have a Verilog netlist, constraints, and SPEF for that block, then use the `createILMDataDir` command to store data in the ILM format.

  Following is the usage of the `createILMDataDir` command:

```
createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -verilog
myfile.v

createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -incr \
-spef max.spef.gz -rcCorner rcMax

createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -incr \
-spef typ.spef.gz -rcCorner rcTyp

createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -incr \
-spef min.spef.gz -rcCorner rcMin

createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -incr \
-sdc funMaxMax.sdc -viewName funct-devSlow-rcMax

createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -incr \
-sdc funMaxTyp.sdc -viewName funct-devSlow-rcTyp

createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -incr \
-sdc tstMaxMax.sdc -viewName test-devSlow-rcMax

createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -incr \
-sdc funMinMin.sdc -viewName funct-devFast-rcMin

createILMDataDir -cts -si -dir block_A.ILM -cell block_A -mmmc -incr \
-sdc tstMinMin.sdc -viewName test-devFast-rcMin
```

createInterfaceLogic

```
createInterfaceLogic
     -dir dirName
     [-hold]
     [-keepSelected]
     [-keepSideCapInst]
     [-noInterClockPath]
     [-noCTS]
     [-noSI]
     [-writeSDC]
     [-ignorePorts ListOfPorts]
```

Creates the specified directory containing ILM files with the appropriate level of extraction for each category: CTS, SI and other (pre-CTS, post-CTS, and post-route). When `-hold` is specified, the command automatically creates both setup and hold ILM models.

The `createInterfaceLogic` command creates a reduced set of instances representing only the interface paths, that is, paths from the I/O ports of the design to the first level of sequential instances (latches are considered sequential elements).

The command automatically modifies the module name with the following rule:

```
topModuleName+timestampl+$+moduleName
```

For more information, see “Creating ILMs for Shared Modules” in the “Using Interface Logic Models in Hierarchical Design” chapter of the Encounter User Guide.

Parameters

`-dir` *`dirName`*	Specifies the location of the generated ILM files.
`-hold`	Creates setup and hold ILM data. You do not need to specify `setAnalysisMode` before you run this command. When you set this parameter, the command `optDesign -hold` uses the data for set and hold fixing.
`-ignorePorts` *`ListOfPorts`*	The given ports and the associated interface path(s) which are starting or ending at these ports will not be included in the ILM model.
`-keepSelected`	

	Preserves all selected instances and nets (in addition to those normally kept) when you create ILMs. Selected instances are those instances that drive the selected nets. You can select instances or nets using the `selectInst` or `selectNet` commands, then use `createInterfaceLogic -keepSelected`. By default, no additional instances and nets are preserved.
`-keepSideCapInst`	Preserves instances that are not on the interface logic path (on a side path) if they connect to instances on the interface logic path. Using this parameter improves the accuracy of OCV analysis, in which the tool finds the early and late paths in the design. Because the side path instances are retained, the tool can find the instance's min and max pin capacitance values in the instance's library cell, and calculate the early and late paths based on these two values.
`-noInterClockPath`	Excludes inter-clock paths from the generated ILM.
`-noCTS`	Does not create a CTS model.
`-noSI`	Does not create an SI model.
`-writeSDC`	Writes SDC constraints based only on the logic kept in the ILM model after the netlist has been trimmed. You can use this feature for ILM validation or for bottom-up SDC promotion: you generate a flat SDC file for top-level implementation analysis.

Examples

- The following command generates ILM files in directory `BlockB`:

  ```
  createInterfaceLogic -dir BlockB
  ```

- The following command creates setup and hold data and stores the ILM files in the `BlockA` directory:

  ```
  createInterfaceLogic -hold -dir BlockA
  ```

Note: In the current ILM flow, SDC constraints originally specified against the complete flat netlist for the design do not get filtered when being applied against the ILM view of the design. Constraints that reference parts of the design that were pruned cause warnings and errors during constraint loading.

You can set the timing_suppress_ilm_constraint_mismatches global variable to true to suppress error and warning messages related to objects not found when loading the SDC constraint files.

Note: The setIlmMode command settings also affect the instances that createInterfaceLogic command will preserve to model the block.

Sample Summary Report

The following is a sample summary report generated at the end of the createInterfaceLogic command:

```
-------------------------------------------------------------------
                createInterfaceLogicSummary
-------------------------------------------------------------------
Model               Reduced Instances         Reduced Registers

ilm_data            7153/7621 (93%)              174/285 (61%)
cts_data            7254/7621 (95%)              0/285 (0%)
si_ilm_data         6793/7621 (89%)              160/285 (56%)
-------------------------------------------------------------------
```

In this report, the reduction ratio in the ilm_data model is 93 percent which means that 7153 out the total 7621 instances for this block have been eliminated. Only 468 instances are written to the Verilog netlist for the ilm_data model out of which 111 instances are registers.

This summary report applies to a block using MMMC. Therefore, views with worst reduction ratio are displayed for each model.

Note: You can run the following commands for improving the reduction ratio:

- setIlmMode -highFanoutPort false
- createInterfaceLogic -noInterClockPath

createNetGroup

```
createNetGroup
     netGroupName
     [-net {netName | netNameList}]
     [-spacing minSpace]
     [-optimizeOrder]
     [-alternateLayer]
```

Creates an empty net group. A net group can contain net names, bus names, or segments of a bus. To add nets into the net group, use the addNetToNetGroup command.

Note: If a net is part of a net group, and you add the same net to another net group, the Encounter software will display a warning, but the net will be added to the second net group too. However, when pin assignment is performed, the net is considered part only of the *first* net group to which it was added—and *not* part of any other net group to which it was subsequently added.

Parameters

`-alternateLayer`	Specifies that pins in the group will be assigned to alternate layers.
`-net {`*`netName	netNameList`*`}`	Specifies the name of the net(s). You can use wildcards (*?) with this parameter.
`netGroupName`	Specifies the name of the created net group name.
`-optimizeOrder`	Specifies that pins in the group are reordered to optimize wire length. If this option is not selected, the pin order is exactly as specified in the pin group.
`-spacing` *`minSpace`*	Specifies the minimum spacing between the pins of the net group. *Default:* The default value is two tracks.

Command Order

You can use this command after importing the design.

Example

The following command creates a net group `bundle1`:

```
createNetGroup bundle1
```

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - "Setting Pin Constraints"
    - "Net Group"

createPinBlkg

```
createPinBlkg
     {-area llx lly urx ury | -edge edgeNumber}
     [-name blockageName]
     [-cell cellName]
     [-layer {layerId | layerIdList}]
```

Creates a pin blockage for a module. You can run this command on the top-level module also.

You can use this command during partition floorplanning.

Parameters

`-area llx lly urx ury`

Specifies the lower left x coordinate, lower left y coordinate, upper right x coordinate, and upper right y coordinate respectively of the blockage area.

Note: The values of *llx*, *lly*, *urx*, and *ury* are relative to the top-level coordinates.

`-cell cellName` Specifies the module for which the pin blockage will be created. This parameter is required if the module is not the top-level module.

`-edge edgeNumber` Specifies the integer value where edge numbering starts from lower-left corner of a partition clock-wise. Edge `0` is the edge that has the smallest y value.

`-layer {layerId | layerIdList}`

Specifies the metal layer(s) for which pins are not to be created.

Default: If no layer number is specified, all metal layers are blocked.

`-name blockageName`

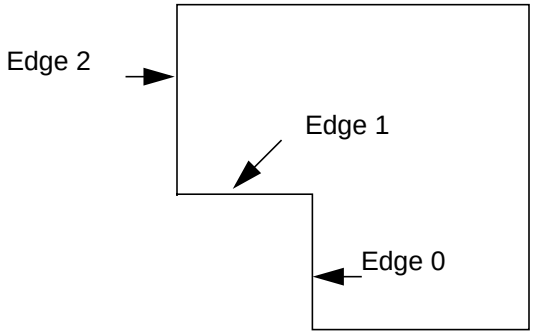
Specifies the name of the partition pin blockage.

Default: By default, the name is `defPtnPinBlkName`.

Examples

- The following command creates a pin blockage named `pinBlkg1` on the edge `0` of layers `M5` and `M7` for the module `PtnBlock1`.

```
createPinBlkg -edge 0 -name pinBlkg1 -cell PtnBlock1 -layer {5 7}
```



- The following command creates a pin blockage named `pinBlkg2` on layers M5 and M7 for the module `PtnBlock2`. The b lockage area is bounded by the lower-left coordinates `100`, `200` and the upper-right coordinates `300`, `400`.

```
createPinBlkg -area 100 200 300 400 -name pinBlkg2 -cell PtnBlock2 -layer {5 7}
```

createPtnCut

```
createPtnCut
     [-ptn partitionName]
     llx lly urx ury
```

Creates a partition rectangular cut from a partition. This object can be created after specifying a module guide to be a partition. Rectangular cuts can only occur on the edge of the partition guide.

You can use this command after specifying partition.

Parameters

llx	Specifies the lower left x coordinate of the cut area.
lly	Specifies the lower left y coordinate of the cut area.
-ptn *partitionName*	Specifies a partition to create a partition cut. If this option is not selected, the command loops through all the partitions an attempts to find the overlapped partition, then creates a partition cut for that partition.
urx	Specifies the upper right x coordinate of the cut area.
ury	Specifies the upper right y coordinate of the cut area.

Example

The following command creates a partition cut area to modify a rectangular partition:

```
createPtnCut 2353.8000 7123.000 2653.8000 8298.2000
```

createPtnFeedthrough

```
createPtnFeedthrough
     llx lly urx ury
     layerId
     -name ptnFeedName
```

Creates a partition routing feedthrough object. The object area keeps pins from being created on the specified metal layer. A cdump file with routing obstruction is created for the top level partition (done when saving the partition).

You can use this command during partition floorplanning.

Parameters

layerId	Specifies the metal layer where pins are not to be created.
llx	Specifies the lower left x coordinate of the feedthrough area.
lly	Specifies the lower left y coordinate of the feedthrough area.
-name *ptnFeedName*	Specifies the name of the routing feedthrough.
urx	Specifies the upper right x coordinate of the feedthrough area.
ury	Specifies the upper right y coordinate of the feedthrough area.

Example

The following command creates a feedthrough object to block pins being generated on layer *M5*:

```
createPtnFeedthrough 2353.8000 7123.000 2653.8000 8298.2000 5
```

createPinGuide

```
createPinGuide
     {-area llx lly urx ury | -edge edgeNumber}
     {-pin pinName | -pinGroup pinGroupName |
          -net {netName | busName} | -netGroup netGroupName | -name objectName}
     [-cell cellName]
     [-layer {layerId | layerIdList}]
```

Creates a pin guide object. The pin guide restricts the pins associated with the pin guide within the specified area. You can run this command on the top-level module also. You can assign the same pin group or net group to multiple pin guides.

Pins are generated for buses, nets, net groups, pins, and pin groups. For buses and nets that do not physically connect to the partition, pins will not be created. Instead, a pin space will be created.

You can use this command during partition floorplanning.

`-area llx lly urx ury`

Specifies the coordinates of the pin guide area.

llx	Specifies the lower left x coordinate of the pin guide area.
lly	Specifies the lower left y coordinate of the pin guide area.
urx	Specifies the upper right x coordinate of the pin guide area.
ury	Specifies the upper right y coordinate of the pin guide area.

`-cell cellName` Specifies the name of the cell. This parameter is not required if you are using the command for a top-level module.

`-edge edgeNumber` Specifies the integer value where edge numbering starts from lower-left corner of a partition clock-wise. Edge `0` is the edge that has the smallest y value.

`-layer {layerId | layerIdList}`

Specifies the metal layer(s) for the pin guide.

Default: If you do not specify the layer number, pin guides are created only on the preferred layers.

Note: Pins will *not* be assigned on non-preferred layers or on the `M1` layer even if the pin guide includes non-preferred layers or the `M1` layer.

`{-pin` *`pinName`* `| -pinGroup` *`pinGroupName`* `| -net {`*`netName`* `|` *`busName`*`} | -netGroup` *`netGroupName`* `| -name` *`objectName`*`}`

`-name` *`objectName`*

Specifies the name of the partition pin guide object. If you specify this parameter, the command will automatically detect whether the object is a pin, pin group, net, net group, or bus. In case there are different objects with the same name, the command selects the object with the following precedence:

- If a cell name is specified, the command selects the object with the following precedence:
  1. **a.** pin
  2. **b.** pin group
  3. **c.** net
  4. **d.** net group or bus
- If a cell name is not specified, the command selects the object with the following precedence:
  1. **a.** net
  2. **b.** net group or bus

For example, if the value of *`objectName`* is `P1`, and there is a pin group named `P1` as well as a bus named `P1`, the command will select the pin group named `P1`.

`-net {`*`netName`* `|` *`busName`*`}`	For nets, specifies the name of the net. For buses, specifies the name of the bus or the bus radix.
`-netGroup` *`netGroupName`*	Specifies the name of the net group.
`-pin` *`pinName`*	Specifies the name of the pin.
`-pinGroup` *`pinGroupName`*	Specifies the name of the pin group.

Default: By default the name is `defRGuideName`.

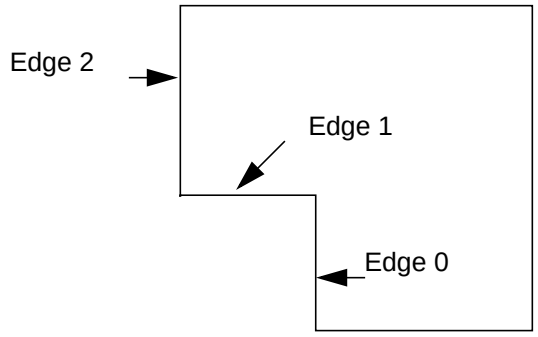
Example

- The following command creates a pin guide object for bus `addr`:

  ```
  createPinGuide -area 2353.8000 7123.000 2653.8000 8298.2000 -name addr
  ```

- The following command creates a pin guide object for net `net1`

  ```
  createPinGuide -edge 2 -cell tdsp_core -net net1
  ```



Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - "Setting Pin Constraints"
    - "Pin Guides"

createPinGroup

```
createPinGroup
     pinGroupName
     [-cell cellName]
     [-pin {pinName | pinNameList}]
     [-spacing minSpace]
     [-optimizeOrder]
     [-alternateLayer]
```

Creates a pin group for a module that is to become a partition. Using this pin group in conjunction with a partition pin guide object, the partition pins are generated. The pins that make up the pin group are the port names of the module.

You use this command during partition floorplanning.

Parameters

`-alternateLayer`	Specifies that pins in the group will be assigned to alternate layers.
`-cell` *`cellName`*	Specifies the name of a module or submodule. This parameter is not required if the command is run for the top-level module.
`-optimizeOrder`	Specifies that pins in the group are reordered to optimize wire length. If this option is not selected, the pin order is exactly as specified in the pin group.
`-pin {`*`pinName`* `|` *`pinNameList`*`}`	Specifies the pin(s) that should be added to the pin group. You can later add more pins to group with the `addPinToPinGroup` command.
`-spacing` *`minSpace`*	Specifies the minimum spacing between the pins of the pin group, in number of tracks or pitch. *Default:* The default value is two tracks.
`pinGroupName`	Specifies the name of the pin group.

Example

The following command creates a pin group `PG_A22` for module `SH17` and adds the pins `pinA22_1` and `pinA22_2` to it.

```
createPinGroup PG_A22 -cell SH17 -pin {pinA22_1 pinA22_2}
```

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - "Setting Pin Constraints"
    - "Pin Group"

createVirtualPartition

```
createVirtualPartition
     -type flatTop
```

Trims the timing graph to ignore logic inside the first level registers in partitions to improve runtime for timing-related commands.

Parameters

`-type flatTop`	Marks the top-level timing graph to mask all logic inside the interface logic of each partition. `flatTop` is the only type you can specify. *Default:* If you do not specify this parameter, timing-related commands must operate on the entire flat netlist rather than on the virtual partition version.

Examples

For examples and more information, see “Chip-Level Budgets Derived by Using Virtual Partitioning” in the Timing Budgeting chapter of the Encounter User Guide, and “Using Virtual Partitioning for Chip-Level Interface Circuit Timing Closure” in the Timing Optimization chapter of the Encounter User Guide.

Note: The `createVirtualPartition` command is now obsolete and has been replaced by `createActiveLogicView`. This obsolete command works in this release, but to avoid warnings and ensure compatibility with future releases, update your script to use `createActiveLogicView`.

definePartition

```
definePartition
     -hinst hInstName
     [-coreSpacing {left right top bottom}]
     [-railwidth railWidth]
     [-minPitchLeft x]
     [-minPitchRight x]
     [-minPitchTop x]
     [-minPitchBottom x]
     [-pinLayerTop {list_of_layers}]
     [-pinLayerBottom {list_of_layers}]
     [-pinLayerLeft {list_of_layers}]
     [-pinLayerRight {list_of_layers}]
     [-reservedLayer {list_of_layers}]
     [-placementHalo {left right top bottom}]
     [-routingHalo integerValue]
     [-routingHaloTopLayer integerValue]
     [-routingHaloBottomLayer integerValue]
     [-stdCellHeight x]
     [-routingStyle {Manhattan | X}]
```

Defines a new partition corresponding to a hierarchical instance.

You can use this command after importing the design.

Parameters

`-coreSpacing {left right top bottom}`

Specifies the space, in micrometers, between the module boundary and core design area of the partition module on the left, right, top, and bottom sides respectively. The partition pins will be located at the partition instance boundary and the core area.

Note: These core-to-edge values are always synchronized between the partition and the corresponding power domain as follows:

If the specified instance was also earlier defined as a power domain, these values for the partition, by default, will have the same value as the values for the power domain. If you specify different values for the partition, the values for the power domain are updated so that they are the same as the values defined for the partition.

Similarly, if a power domain is created (or an existing power domain modified) on the specified instance *after* the partition is defined, the values set earlier for the partition are updated so that they are the same as the values defined for the power domain.

`-hinst` *`hinstName`*
Specifies the name of the hierarchical instance.

Note: None of its ancestors or descendants in the netlist tree can be a partition.

`-minPitchLeft` *`x`*
`-minPitchRight` *`x`*
`-minPitchTop` *`x`*
`-minPitchBottom` *`x`*
Specifies the pin pitch dimension for the *Left*, *Right*, *Top*, and *Bottom* partition sides. The default is 2, which places one pin for every two metal tracks. The actual pin pitch depends on the pin's metal layer.

The Partition Pin Guide floorplan object will supersede this entry and use the *Min. Space* entry in the object's Attribute Editor.

`-pinLayerTop {`*`list_of_layers`*`}`
`-pinLayerBot {`*`list_of_layers`*`}`
`-pinLayerLeft {`*`list_of_layers`*`}`
`-pinLayerRight {`*`list_of_layers`*`}`

Specifies the metal layers to use for the partition. The metal layers correspond to the sides of the partition, where the vertical metal layers (for example, *M2*, *M4*, and so on) are for the Top and Bottom sides, and horizontal metal layers (for example, *M3*, *M5*, and so on) are for the Left and Right sides of the partition.

Specify this value in integers, for example `-pinLayerTop {2 4}`

`-placementHalo{`*`left right top bottom`*`}`

Specifies extra spacing, in micrometers (around the left, bottom, right and top sides respectively) of the partition that should not be used for placement. The placement program will not place standard cells in the area for partition placement. This spacing is called *placement halo.*

At the top-level design, this information is saved as part of the partition section in a floorplan file. This information is also saved in a partition floorplan file when saving partitions. By default, the value is 0 for all the sides.

If the hierarchical instance was earlier defined as a power domain, then this field corresponds to the minimum gap parameter of the power domain.

Note: The minimum gap value of the power domain and the placement halo values of the partition are always synchronized as follows:

If the specified instance was also earlier defined as a power domain, the placement halo values for the partition, by default, will have the same value as the minimum gap values for the power domain. If you specify a different value for the partition, the value for the power domain is updated so that it is the same as the value defined for the partition.

Similarly, if a power domain is created (or an existing power domain modified) on the specified instance *after* the partition is defined, the value set earlier for the partition is updated so that it is the same as the value defined for the power domain.

`-railwidth` *`railWidth`*

Specifies the standard cell power rail width, in micrometers. If partition pins are not allowed above power or ground rail width, specify the dimension of the power rail width. If a value of 0.0 is entered (the default), the width from the Technology file will be used.

`-reservedLayer {`*`list_of_layers`*`}`

Specifies the metal layers that are used for routing in the partition and generating partition pins. Any metal layers that are not specified, usually the top-most metal layers, are allowed to route over the partition.

A normal six-metal layer specification is `M1`, `M2`, `M3`, `M4` and `M5` specified, and `M6` unspecified. When saving the partition, the LEF generated for this partition will have routing blockages on their layers so that the top-level router is aware of which metal layers are being used in the partition.

`-routingHalo` *`integerValue`*

Specifies routing spacing, in micrometers, around the sides of the partition.

A routing halo is honored only by the signal router. The signal router treats routing on the specific routing layers in the routing halo area as very high cost routing. However, perpendicular routing or straight connections to pins are acceptable. The special router does not honor routing halos. To block special routing, use routing blockages instead.

You can specify a positive or a negative value for the routing halo. A positive value means that the halo will be outside the partition. A negative value means that the halo will be applicable during block implementation, inside of the partition.

The same value is used for all sides of the partition.

Note: For instance blocks— blackboxes, hard macros or block-level designs—specify the routing halo through the *Floorplan – Edit Floorplan – Edit Halos* form in the GUI or through the `addRoutingHalo` command. Also, for bottom- up hierarchal flow, specify routing halo for a block at the top-level design or at the block-level design through the *Floorplan – Edit Floorplan – Edit Halos* form in the GUI or through the `addRoutingHalo` command.

`-routingHaloTopLayer` *`integerValue`*

Specifies the top partition layer for which routing halo will be created.

Default: By default, the top layer reserved for the partition is used.

`-routingHaloBottomLayer` *`integerValue`*

Specifies the bottom partition layer for which routing halo will be created.

Default: By default, the bottom layer reserved for the partition is used.

`-routingStyle {Manhattan | X}`

Specifies whether the routing style is Manhattan (orthogonal) or X.

Default: The default routing style is Manhattan.

`-stdCellHeight` *`x`* Specifies the standard cell height for the partition.

Default: The default value is derived from the default technology site and row height.

Example

```
definePartition -hinst ctr_inst
-coreSpacing 0.56 0.56 0.0 0.0
-railWidth 0.0
-minPitchLeft 2 -minPitchRight 2
-minPitchTop 2 -minPitchBottom 2
-reservedLayer {1 2 3 4}
-pinLayerTop {2 4} -pinLayerLeft {3} -pinLayerBottom {2 4} -pinLayerRight {3}
-placementHalo 10.0 0.0 0.0 0.0 -routingHalo 10.0
-routingHaloTopLayer 7 -routingHaloBottomLayer 1
```

deleteAllPartitions

```
deleteAllPartitions
```

Removes all specified partitions. The partitions are flattened (unpartitioned) before they are removed.

You can use this command after committing a partition.

Parameters

None

deleteAllPtnCuts

`deleteAllPtnCuts`

Deletes all the partition cuts in the floorplan.

You can use this command after specifying a partition.

Parameters

None

deleteAllPtnFeedthroughs

`deleteAllPtnFeedthroughs`

Deletes all the partition feedthrough objects in the floorplan.

You can use this command during partition floorplanning.

Parameters

None

deleteBlackBox

`deleteBlackBox` *`instName`*

Changes the instance from a blackbox to a hard macro.

You can use this command after importing the design.

Parameters

`instName`	Specifies the name of the blackbox instance. You can use wildcards (`*?`) with this parameter.

Example

The following command changes all blackbox instances that start with the letter `A` back to a hard macro:

```
deleteBlackBox A*
```

deleteNetFromNetGroup

```
deleteNetFromNetGroup
     netGroupName
     -net {netName | netNameList}
```

Removes a net from a net group.

You can use this command after importing the design.

Parameters

netGroupName	Specifies the name of the net group. You can use wildcards (*?) with this parameter.
-net {*netName* | *netNameList*}	Specifies the name of the net(s). You can use wildcards (*?) with this parameter.

Example

The following command removes net NET1 from net group NG22:

```
deleteNetFromNetGroup NG22 NET1
```

deleteNetGroup

```
deleteNetGroup
     netGroupName
```

Removes a net group that was created earlier.

Note: When you delete a net group, any bus guide associated with the net group also gets deleted.

Parameters

netGroupName	Specifies the name of the net group. You can use wildcards (*?) with this parameter.

Example

The following command removes netgroup NG_22.

```
deleteNetGroup NG_22
```

deletePartition

`deletePartition` *`ptnName`*

Deletes the specified partition.

Parameters

`ptnName`	Specifies the name of the partition to delete.

Example

The following command deletes the partition `p1`.

```
deletePartition p1
```

deletePinBlkg

```
deletePinBlkg
     {-name pinBlkgName} | -all
```

Deletes a pin blockage.

Parameters

`-all`	Specifies that all pin blockages will be deleted.
`-name` *`pinBlkgName`*	Specifies the name of the pin blockage. You can specify wildcards (?*) with this parameter.

Example

The following command deletes the pin blockage `PtnPinBlkg1`:

```
deletePinBlkg -name PtnPinBlkg1
```

deletePinFromPinGroup

```
deletePinFromPinGroup
     [-cell cellName]
     -pinGroup pinGroupName
     -pin {pinName | pinNameList}
```

Removes a port from a specified pin group.

Parameters

`-cell` *`cellName`*
: Specifies the name of the module. The module name is not required if the command is run on a top-level module.

`-pinGroup` *`pinGroupName`*
: Specifies the name of the pin group that was created.

`-pin {`*`pinName | pinNameList`*`}`
: Specifies the name(s) of the port(s) of the module.

Example

The following command removes ports `RASN1` and `RASN2` of module `SH17` from pin group `pin17`:

```
deletePinFromPinGroup -cell SH17 -pinGroup pin17 -pin {RASN1 RASN2}
```

deletePinGroup

```
deletePinGroup
     [-cell cellName]
     -pinGroup pinGroupName
```

Deletes a pin group that was created earlier.

Parameters

`-cell` *`cellName`*
: Specifies the name of the module. The module name is not required if the command is run on a top-level module.

`-pinGroup` *`pinGroupName`*
: Specifies the name of the pin group that was created.

Example

The following command deletes the pin group `PinGroup17` of module `SH17`:

```
deletePinGroup -cell SH17 -pinGroup PinGroup17
```

deletePinGuide

```
deletePinGuide
     [-cell cellName]
     {-pin pinName | -pinGroup pinGroupName |
          -net {netName | busName} | -netGroup netGroupName | -name objectName}
     | -all
```

Deletes a pin guide object in the floorplan.

You can use this command during partition floorplanning.

Parameters

`-all`	Specifies that all pin guides will be deleted.
`-cell cellName`	Specifies the name of the module. The module name is not required if the command is run on a top-level module.

`{-pin pinName | -pinGroup pinGroupName | -net {netName | busName} | -netGroup netGroupName | -name objectName}`

`-name` *`objectName`*

Specifies the name of the partition pin guide object. If you specify this parameter, the command will automatically detect whether the object is a pin, pin group, net, net group, or bus. In case there are different objects with the same name, the command selects the object with the following precedence:

- If a cell name is specified, the command selects the object with the following precedence:
  1. **a.** pin
  2. **b.** pin group
  3. **c.** net
  4. **d.** net group or bus
- If a cell name is not specified, the command selects the object with the following precedence:
  1. **a.** net
  2. **b.** net group or bus

For example, if the value of *`objectName`* is `P1`, and there is a pin group named `P1` as well as a bus named `P1`, the command will select the pin group named `P1`.

`-net {`*`netName`* `|` *`busName`*`}`

For nets, specifies the name of the net. For buses, specifies the name of the bus or the bus radix.

`-netGroup` *`netGroupName`*

Specifies the name of the net group.

`-pin` *`pinName`*

Specifies the name of the pin.

`-pinGroup` *`pinGroupName`*

Specifies the name of the pin group.

Default: By default the name is `defRGuideName`.

Example

The following command deletes a pin guide object named `group_1`:

```
deletePinGuide -name group_1
```

deletePtnAllPtnCuts

`deletePtnAllPtnCuts` *`ptnName`*

Deletes all the partition cuts for a partition.

Parameters

`ptnName`	Specifies the name of the partition.

Example

The following command deletes all partition cut objects for partition `sheet17`:

```
deletePtnAllPtnCuts sheet17
```

editPin

```
editPin
     [-cell cellName]
     -pin {pinName | pinNameList}
     {-spreadType {START | CENTER | SIDE | EDGE | RANGE} |
          -assign x y}
     [{-side sideName} | {-edge edgeNumber}]
     [-layer layerId]
     [-use {SIGNAL | CLOCK | ANALOG}]
     [-fixOverlap {0 | 1} [-honorConstraint {0 | 1}]]
     [-snap {TRACK | USERGRID | MGRID}]
     [-pinWidth pinWidthValue]
     [-pinDepth pinDepthValue]
     [-start x y]
     [-end x y]
     [-spacing spacingValue]
     [-unit {MICRON | TRACK}]
     [-fixedPin {0 | 1}]
```

Modifies properties of pins, such as pin spreading, pin location, pin width and depth, spacing, snap-to location, and status. The `editPin` command provides the equivalent capabilities of the Pin Editor form.

Note: The `editPin` command does not support wildcards.

The `editPin` command honors the minimum area rule as follows:

- If neither the `-pinWidth` nor the `-pinDepth` parameters have been specified, the `editPin` command takes the minimum pin width and the minimum area from the LEF file and calculates the pin depth accordingly.
- If only the `-pinWidth` parameter has been specified, the `editPin` command takes the minimum area from the LEF file and calculates the pin depth accordingly.
- If only the `-pinDepth` parameter has been specified, the `editPin` command takes the minimum area from the LEF file and calculates the pin width accordingly.

If both the `-pinWidth` and the `-pinDepth` parameters are specified, the `editPin` command takes these specified values, without enforcing the minimum area rule.

Note: The value of the parameters of the `editPin` command is not stored across multiple runs of the command—you need to specify the required value with every run of the command. For example, if you do not specify the `-pinWidth` parameter, the default pin width of the specified layer is used.

Parameters

`-assign {`*`x y`*`}`

Preassigns a pin at the specified location. You can use this parameter with other parameters such as `-snap`, `-fixOverlap`,`-side`, and so on.

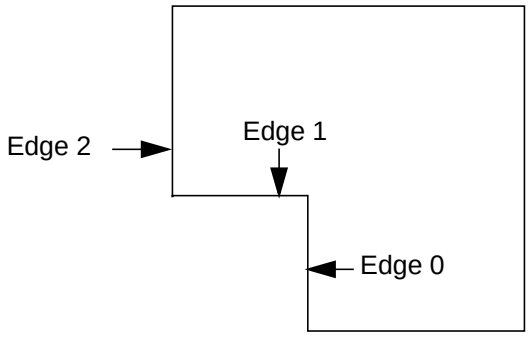
`-cell` *`cellName`*

Specifies the name of the module.

If you do not specify a cell name, the command runs on the top-level cell.

`-edge` *`edgeNumber`*

For rectilinear partitions, specifies the edge along which the pins will be placed. The edge number is an integer; the edge numbering starts from lower-left corner of a partition clock-wise. Edge `0` is the edge that has the smallest y value.



To spread pins on multiple rectilinear edges that are on the same side, use the `-side` parameter

For rectangular partitions, you can specify a side with the `-side` parameter.

`-end` *`x y`*

For pin spreading along a range, specifies the coordinates where pins should end. This parameter is used with the `-spread` *`RANGE`* parameter.

`-fixedPin {0 | 1}`

A value of 1 sets the pin status to *Fixed.*

Default: The pin status is *placed*.

`-fixOverlap {0 | 1}`

A value of 1 specifies that the pins should be moved to the closest acceptable position such that pins do not overlap.

Default: Pin overlapping is on.

This parameter applies only when pins are snapped to layer track.

`-honorConstraint {0 |1}`

Specifies whether user-defined pin constraints should be honored.

The following constraints are honored when this parameter is set to on: pin spacing, pin size, pin-to-corner distance, and partition-allowed pin layers.

Default: `0` (user-defined pin constraints are not honored)

`-layer` Specifies the layer on which the pins will be assigned.

`-pin {`*`pinName`* `|` *`pinNameList`*`}`

Specifies the pins, or the list of pins of the specified module on which the command will run.

`-pinDepth` Sets the depth of the pin in microns.

Default: If you do not specify this parameter, the default depth of the specified layer is used.

`-pinWidth` Sets the width of the pin in microns.

Default: If you do not specify this parameter, the default width of the specified layer is used.

`-side` *`sideName`*

For rectangular partitions, specifies the side on which the pins will be placed. The values for *`sideName`* can be: `TOP`, `BOTTOM`, `LEFT`, `RIGHT`, or `INSIDE`.

Note: The value `INSIDE` can only be specified with the `-assign` parameter.

For rectilinear partitions, you can also specify an edge with the `-edge` parameter.

`-snap {TRACK | USERGRID | MGRID}`

Specifies how the pins will be snapped.

`TRACK`	Snaps pins to layer tracks.
`USERGRID`	Snaps pins to the user-defined grid.
`MGRID`	Snaps pins to the manufacturing grid.

Default: The default value is `TRACK`.

`-spacing` *`spacingValue`*

Displays the spacing between pins for multi-pin editing. The spacing is either in micrometers or by layer track, as specified by the `-unit` parameter.

Default: The spacing is set automatically.

`-spreadType {START | CENTER | SIDE | EDGE | RANGE}`

Specifies how to spread pins along a block's edge.

`START`	Uses the coordinates of the first pin specified with the `-pin` parameter as the starting point (anchor) for spreading a group of pins. This value can be used even if you specify a single pin with the `-pin` parameter.
`CENTER`	If a single pin is specified with the `-pin` parameter, the pin will be placed at the center of the specified edge/side. If multiple pins are specified, the pin editor automatically calculates the starting and end points such that the center pin(s) of the group are placed at the center of the specified edge or side.
`SIDE`	If you specify a side with the `-side` parameter and specify the `-spreadType SIDE` parameter, the pins are spread evenly along all the edges belonging to the specified side. The pins are spread from left to right and from bottom to top.

	`EDGE`	If you specify an edge with the -edge parameter and specify the `-spreadType` `EDGE` parameter, the pins are spread evenly along the entire specified edge, using the limits of the edge as the starting and ending points.
	`RANGE`	Spreads pins evenly along the block edge between points you specify with `-start` {*x y*} and `-end` {*x y*} parameters. The ending coordinates become the coordinates of the last pin in the pin group. The Pin Editor determines the appropriate spacing. For rectilinear partitions, if the start and/or end points are not on the specified side, the start and/or end points are snapped to the nearest correct location on the specified side.
`-start {`*x y*`}`		For pin spreading along a range, specifies the coordinates where pins should start. This parameter is used with the `-spreadType` `RANGE` and the `-spreadType` `START` parameters.

`-unit {MICRON | TRACK}`

Specifies the unit in microns or in tracks. This parameter applies only to the unit of the `-spacing` parameter.

Default: Microns

Related Topics

- "Floorplanning the Design" chapter in the *Encounter User Guide*
  - Using the Pin Editor
- Partitioning the Design chapter in the *Encounter User Guide*
  - Assigning Pins
- "Edit Menu" chapter in the *Encounter Menu Reference*
  - Pin Editor

Example

The following command modifies the pins as specified here:

- the command runs on the module `DTMF_CORE_INST1`
- the command runs on all pins whose name starts with `DTMF_CORE_INST1_OUTPIN`
- the pins are spread on the edge number `2`
- the USE property is set to `CLOCK`
- the pins are spread between the coordinates (`17 23`) and (`117 123`)
- the pins are assigned on the layer `M3`
- the pin status is set to *Fixed*
- the pins are snapped to layer tracks
- the unit is in tracks
- the spacing between pins for multi-editing is set to 2 tracks
- the pin depth is set to 10 microns.

  Note: The `-unit` parameter is set to `TRACK` but that does not affect the unit of pin depth.

- the pins are moved to the nearest acceptable location as the `-fixOverlap` parameter is set to 1. However, the `-overlapConstraints` parameter is not specified.

```
editPin
-cell DTMF_CORE_INST1
-pin {DTMF_CORE_INST1_OUTPIN}
-spreadType RANGE
-edge 2
-layer 3
-use CLOCK
-fixOverlap 1
-snap TRACK
-pinDepth 10
-start {17 23}
-end {117 23}
-spacing 2
-unit TRACK
-fixedPin 1
```

estimatePtnChannel

```
estimatePtnChannel
     reportFileName
     [-adjustModule]
     [-reportOnly]
     [-utilize factor]
     [-utilizeFile fileName]
     [-resize]
```

Estimates the required spacing between partitions, black boxes, and hard macros for a given floorplan. It automatically adjusts the partitions, black boxes, and hard macros based on the required spacing, and adjusts the die and core area if necessary. This command also reports the current and required spacing, in micrometers, between a pair of objects (partition, black box, hard macro, or core boundary).

This command can be used after floorplanning, specifying the partition, or running the `partition` command.

Note: The `estimatePtnChannel` command does not support designs with a master partition and its partition clones. Before running this command, you must uniquify the netlist using the `uniquifyNetlist` command.

Note: For rectilinear fences, the `estimatePtnChannel` command estimates the spacing based on the coordinates of the rectangular bounding box of the fence.

Parameters

`-adjustModule`

Includes the adjustment of modules that are not specified as partitions, and groups (guides, regions, and fences).

For each module or group that needs to be adjusted, move it so that the original minimum spacing between this object and the adjacent objects is maintained.

`reportFileName`

Specifies the report output file, if any.

If a report output file exists, this must be the first parameter specified with the command.

Default: If you do not specify this parameter, the filename is *`topCellName`*`.channel`.

`-reportOnly` Generates a report without adjusting the floorplan.

Note: When you run the `estimatePtnChannel` command with the `-reportOnly` option, you can view the violated distance between blocks in the GUI. For this, select the *Channel Congestion* label in the *All Colors* form of the GUI.

`-resize` Automatically reshapes module guides, fences, or blackboxes to improve congestion. If the user-specified block aspect ratio constraints are available, the channel width estimator honors them; otherwise, it uses the default global aspect ratio.

`-utilize` *`factor`* Specifies an extra margin percentage for the channel spacing. The factor is a real number between 0 (zero percent) and 1 (100 percent). For example, if the required channel width is 10 micrometers and a utilization factor is .5 (50%), then the final channel width is 20 micrometers.

Default: 1

`-utilizeFile` *`fileName`*

Specifies a channel utilization between two given blocks or partitions.

flattenCoverCell

```
flattenCoverCell
     [-inst {instanceName | instanceNameList}]
```

Dissolves and flattens any instance whose master has class COVER, but no subclass BUMP. You can use this command after loading a DEF file (defIn).

Parameters

`-inst {instanceName | instanceNameList}`

Specifies the instance, or a list of instances, to be flattened. If you are specifying a list, separate the instance names with a space.

Default: If you do not specify this parameter, all instances whose masters have class COVER are flattened.

Examples

- The following command flattens the cover cells named coverCell1 and coverCell2

  ```
  flattenCoverCell -inst {coverCell1 coverCell2}
  ```

- The following command flattens *all* cover cells

  ```
  flattenCoverCell
  ```

flattenIlm

`flattenIlm`

Flattens the interface logic models (ILMs) at the top level so the entire design can be analyzed at the top. When you use this command, the software switches from blackbox view to ILM view.

If you do not run `flattenIlm`, the ILM netlist will not be visible to the Encounter software. You can switch between the ILM view and the blackbox view using the `flattenIlm` and `unflattenIlm` commands.

You can use this command after running the Placement, Trial Route, and Extract RC programs.

Parameters

None

flattenPartition

```
flattenPartition
     [-noUninheritPhysical]
     [-bringBackRow]
     partitionName ...
```

Uncommits (flattens) the partition back to normal module status.

This command should also be used to unassign blackbox pins *after* the blackbox is committed. To unassign blackbox pins *before* the blackbox is committed, use the `setPtnPinStatus` command. This command unassigns partition pins as well.

To bring back row information for a partition, use the `-bringBackRow` parameter. If the partition rows are non-integer-multiple-height rows or are not aligned with the top-level rows, the `flattenPartition` command will automatically create a power domain associated with the partition. The power domain will have the following characteristics:

- The timing library will the same as that for the top level.
- If a global net connection is defined in the partition block, the power domain will have the same global net connection. Otherwise the global net connection for the power domain will be set to the top-level net connection.
- The minimum gap values will be the same as the placement halo values for the partition.

The recommended use model for bringing back non-integer-multiple height rows or unaligned rows is as follows.

Top-down flow:

1. Create a power domain at full chip-level design.
2. Specify the same hierarchical instance as the partition

Bottom-up flow:

1. Create a power domain at top-level design.
2. Assign the instance block as a member of the power domain created in step 1.

Parameters

`-bringBackRow`	Specifies that partition row information should be brought back for cases where: ■ the partition is not a power domain and row information in partition is different from top-level design, *and* ■ after uncommiting the partition, the rows of the partition space are not aligned with the top-level rows When you use this parameter, a power domain is created for the partition. *Default:* If you do not specify this parameter, the row information is *not* brought back in such cases.
`-noUninheritPhysical`	Specifies that physical only cell instances are removed during `flattenPartition`.
`partitionName ...`	Specifies the name of the partition. You can specify any number of partitions.

Example

The following command uncommits the partition status of partitions `sheet17` and `sheet25`:

```
flattenPartition sheet17 sheet25
```

getActiveLogicViewMode

```
getActiveLogicViewMode
     [-help]
     [-async {false | true}]
     [-highFanoutPort {true| false}]
     [-loopBack {false | true}]
     [-quiet]
```

Controls certain behaviors of active logic view commands.

Use the `getActiveLogicViewMode` command to display the current settings for the `setActiveLogicViewMode` command.

Parameters

`-async {false | true}`

Discards or retains the portion of the netlist relating to asynchronous control and scan controls terminals. By default, the tool discards this portion of the netlist. This option potentially conflicts the setting of `setAnalysisMode -async true` when the analysis for an asynchronous circuit is desired. When this happens, a warning message is issued.

Default: `false`

`-highFanoutPort {true | false}`

Retains or discards the portion of the netlist relating to high-fanout ports. By default, the tool retains this portion of the netlist.

Default: `true`

`-loopBack {false | true}`

Discards or retains the portion of the netlist relating to input and output loopback paths. By default, the tool discards this portion of the netlist.

Default: `false`

-help	Outputs a brief description that includes type and default information for each getActiveLogicViewMode parameter. For a detailed description of the command and all of its parameters, use the man command: man getActiveLogicViewMode
-quiet	Displays the current settings for the specified parameters in the Tcl list format. If you specify -quiet without any parameters, the software displays the current settings of all the getActiveLogicViewMode parameters in the Tcl format.

Example

- The following command displays the current setting for the highFanoutPort parameter in the Tcl list format:

  ```
  getActiveLogicViewMode -highFanoutPort -quiet
  ```

  The software displays the following information:

  ```
  true
  ```

- The following command displays the current setting for the getActiveLogicViewMode parameter:

  ```
  getActiveLogicViewMode
  ```

  The software displays the following information:

  ```
  async false                    #bool, default=false
  -highFanoutPort true           #bool, default=true
  -loopBack false                #bool, default=false

  {async false}  {highFanoutPort true}   {loopBack false}
  ```

- The following command displays the current settings for all the getActiveLogicViewMode commands in the Tcl list format only:

  ```
  getActiveLogicViewMode -quiet
  ```

  The software displays the following information:

  ```
  {async false} {highFamoutPort true} {loopBack false}
  ```

getAllowedPinLayersOnEdge

```
getAllowedPinLayersOnEdge
     {-edge edgeName | -all}
     [-cell cellName]
```

Retrieves the pin layers for a specified edge or for all edges.

You can use this command after using the setAllowedPinLayersOnEdge command.

Parameters

`-all`	Specifies that pin layers for all the layers should be retrieved.
`-cell cellName`	Specifies the name of the module for which the layers should be retrieved. *Default:* if you do not specify this parameter, the command is run on the top cell.
`-edge edgeName`	Specifies the name of the edge. This parameter can take the following values: ❑ `N, S, W, E` (supports both upper and lower case): For North, South, West, or, East sides, respectively. ❑ `T, B, L, R` (supports both upper and lower case): for Top, bottom, Left, or Right sides respectively. ❑ `dbcN, dbcS, dbcE, dbcW`

Example

The following command retrieves the pin layers for the top side.

```
getAllowedPinLayersOnEdge -edge T
```

getBlackBoxArea

```
getBlackBoxArea
     -cell cellName
     [-stdCellArea]
     [-macroArea]
     [-cellUtil]
```

Retrieves the standard cell area, macro area, and cell utilization value for the specified blackbox. If only the blackbox cell name is specified, the command reports the total blackbox area.

Important

You can use this command to report the black box area only if you had specified the blackbox using the specifyBlackBox command with the `-gateArea` parameter.

Parameters

`-cell cellName`	Specifies the name of the blackbox cell.
`-cellUtil`	Reports the standard cell utilization value for the blackbox. The standard cell utilization is calculated as follows: standard cell area / (blackbox area - total macro area) This parameter cannot be used if the blackbox was specified based on the total area or the gate count.
`-macroArea`	Reports the area of the macros included in the blackbox area. This parameter cannot be used if the blackbox was specified based on the total area or the gate count.
`-stdCellArea`	Reports the area of the standard cell included in the blackbox area. This parameter cannot be used if the blackbox was specified based on the total area or the gate count.

Examples

- The following command retrieves the total blackbox area for the blackbox `bbox_inst1:`

  ```
  getBlackBoxArea -cell bbox_inst1
  ```

- The following command retrieves the standard cell area, macro area, and cell utilization value for the blackbox `bbox_inst2`:

```
getBlackBoxArea -cell bbox_inst2 -stdCellArea -macroArea -cellUtil
```

getGlobalMinPinSpacing

```
getGlobalMinPinSpacing
```

Retrieves the global minimum pin spacing that was defined with the `setGlobalMinPinSpacing` command. The pin spacing is defined in number of tracks. For example, a minimum pin spacing of 1 means that the minimum distance between pins is 1 track (pins are placed at adjacent tracks).

Parameters

None

getIlmMode

```
getIlmMode
     [-async]
     [-highFanoutPort]
     [-loopBack]
     [-quiet]
```

Displays the following information about a `setIlmMode` parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setIlmMode` parameters.

Parameters

`parameter_names`

Displays information for the specified parameters. You can specify one or more parameters.

See `setIlmMode` for descriptions of the parameters you can specify.

`-quiet`

Displays the current settings for the specified parameters in Tcl list format only.

If you specify `-quiet` without any parameters, the software displays the current settings of all `setILMMode` parameters in Tcl format.

Examples

- The following example displays the current setting for the `-async` parameter:

  ```
  getIlmMode -async
  ```

  The software displays the following information:

  ```
  -async false                    # bool, default=false
  false
  ```

- The following command displays the current settings for all `setIlmMode` parameters:

  ```
  getIlmMode
  ```

  The software displays the following information:

  ```
  -async false                    # bool, default=false
  -highFanoutPort true            # bool, default=true
  -loopBack false                 # bool, default=false

  {async false} {highFanoutPort true} {loopBack false}
  ```

- The following command displays the current setting for the `-highFanoutPort` parameter in Tcl list format only:

  ```
  getIlmMode -highFanoutPort -quiet
  ```

  The software displays the following information:

  ```
  true
  ```

- The following command displays the current settings for all `setIlmMode` parameters in Tcl list format only:

  ```
  getIlmMode -quiet
  ```

  The software displays the following information:

  ```
  {async false} {highFanoutPort true} {loopBack false}
  ```

getIlmType

```
getIlmType
     [-model]
```

Displays the following information about `setIlmType` in the Encounter log file and the Encounter console:

- Parameter name
- Current value
- Type (boolean, string, and so on)
- The current value set by the user

If you do not specify a parameter, the Encounter software displays information for all of the `setIlmType` parameters.

Parameters

`parameterNames`	Displays information for the specified parameters. You can specify one or more parameters. For more information, see the details of the `setIlmType` parameters.

Examples

- The following example displays the current setting for the `-model` parameter:

  ```
  getIlmType -model
  ```

  The software displays the following information:

  ```
  timing
  ```

getLayerPinDepth

```
getLayerPinDepth
     [-cell cellName]
     -layer {layerId | layerIdList}
```

Retrieves the depth of the pins on the specified layer(s) on the specified module. You can run this command on a top-level module also.

You can use this command after using the `setLayerPinDepth` command.

Parameters

`-cell cellName`
: Specifies the name of the module. The module name is not required if the command is run on a top-level module.

`-layer {layerId | layerIdList}`
: Specifies the metal layer(s) for which the pin depth should be retrieved.

Example

The following command retrieves the pin depth on layer `M5` of the module `SH17`.

```
getLayerPinDepth -cell SH17 -layer 5
```

getLayerPinWidth

```
getLayerPinWidth
     [-cell cellName]
     -layer {layerId | layerIdList}
```

Retrieves the width of the pins on the specified layer(s) on the specified module. You can specify a top-level module with this command.

You can use this command after using the `setLayerPinWidth` command.

Parameters

`-cell cellName` Specifies the name of the module. The module name is not required if the command is run on a top-level module.

`-layer {layerId | layerIdList}`

Specifies the metal layer(s) for which the pin width should be retrieved.

Example

The following command retrieves the pin width on layer `M5` of the module `SH17`.

```
getLayerPinWidth -cell SH17 -layer 5
```

getMinPinSpacingOnEdge

```
getMinPinSpacingOnEdge
     [-cell cellName]
     {-edge edgeName | -all}
```

Retrieves the minimum pin spacing, for a specified module and edge(s), that was defined with the setMinPinSpacingOnEdge command. You can specify a top-level module with this command.

Parameters

`-all`	Specifies that the minimum pin spacing should be retrieved for all edges.
`-cell cellName`	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
`-edge edgeName`	Specifies the name of the edge. This parameter can take the following values: ❑ `N, S, W, E` (supports both upper and lower case): For North, South, West, or, East sides, respectively. ❑ `T, B, L, R` (supports both upper and lower case): for Top, bottom, Left, or Right sides respectively. ❑ `dbcN, dbcS, dbcE, dbcW`

Example

The following command retrieves the minimum pin spacing on all edges of the module `SH17`.

```
getMinPinSpacingOnEdge -cell SH17 -all
```

getPinDepth

```
getPinDepth
     [-cell cellName]
     pinName
```

Retrieves the pin depth for a specified pin in the specified module. You can specify a top-level module with this command.

Parameters

`-cell cellName`	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
pinName	Specifies the name of the pin.

Example

The following command retrieves the pin depth of pin `Pin2_PTN1` in module `PTN1.`

```
getPinDepth -cell PTN1 Pin2_PTN1
```

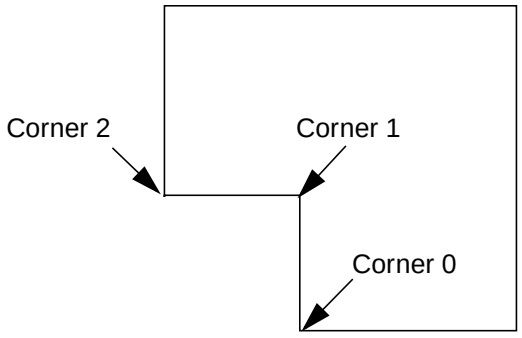
getPinToCornerDistance

```
getPinToCornerDistance
     [-cell cellName]
     {corner cornerNumber |
     -all}
```

Retrieves the pin-to-corner distance that was defined with the `setPinToCornerDistance` command.

Parameters

`-all`	Specifies that the pin-to-corner distance should be retrieved for all corners.
`-cell` *`cellName`*	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
`-corner` *`cornerNumber`*	Specifies the corner of the partition block. This is an integer value, where corner numbering starts at 0 from the lower-left corner of a partition clock-wise. Corner `0` is the corner that has the smallest y value.



Example

The following command retrieves the pin-to-corner distance for corner `0` for the module `PN5`.

```
getPinToCornerDistance -cell PN5 -corner 0
```

getPinWidth

```
getPinWidth
     [-cell cellName]
     pinName
```

Retrieves the pin width for a specified pin in the specified module. You can specify a top-level module with this command.

Parameters

`-cell cellName`	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
pinName	Specifies the name of the pin.

Command Order

You can use this command during partitioning floorplanning.

Example

The following command retrieves the width of pin `Pin2_PTN1` in module `PTN1`

```
getPinWidth -cell PTN1 Pin2_PTN1
```

getPtnPinStatus

```
getPtnPinStatus
     partitionName
     pinName
```

Retrieves the pin status that was defined with the setPtnPinStatus command.

Parameters

partitionName	Specifies the name of the partition.
pinName	Specifies the name of the pin of the partition.

getPtnUnalignedNets

`getPtnUnalignedNets` *`fileName`*

Report the names of the nets where the pins are not aligned for the following cases:

- nets that have two pins on two different partitions and/or clones and are not connected to top level hard-macro, standard cell, or I/O
- nets that have two that are connected to the same partition
- nets that have one pin connected to top level hard-macro, standard cell, or I/O and the other pin connected to a partition or a clone

Note: The nets reported by this command are also highlighted in the artwork window.

Important

Command `getPtnUnalignedNets` is now obsolete and will be removed in the next major release of the software. This command has been replaced by `reportUnalignedNets` command

Parameters

`fileName`	Specifies an output report file.

hiliteFeedthroughNets

```
hiliteFeedthroughNets
     fileName
     [netName | netNameList]
```

Highlights all nets for which feedthrough buffers were inserted with the `insertPtnFeedthrough` command. The highlighted feedthrough path consists of the nets, the terms that the nets connect to, and the instances that contain those terms.

For the `hiliteFeedthroughNets` to work, the `insertPtnFeedthrough` command must be run with the `-netMapping` parameter. The net mapping file generated with the `insertPtnFeedthrough -netMapping` parameter is used by the `hiliteFeedthroughNets` command to highlight the feedthrough nets.

To dehighlight the feedthrough nets, run the `dehighlight` command.

Parameters

`fileName`
Specifies the net mapping file that was generated with the `insertPtnFeedthrough -netMapping` parameter.

`netName | netNameList`
Specifies the name of the net(s) whose feedthrough path should be highlighted.

Default: The feedthrough paths of all nets specified in the net mapping file are highlighted.

Examples

- The following command highlights all feedthrough nets specified in the net mapping file `nmpFile1`.

  ```
  hiliteFeedthroughNets nmpFile1
  ```

- The following command highlights all feedthrough nets that are specified in the net mapping file `nmpFile1` whose name begins with `feedthruNet_core`.

  ```
  hiliteFeedthroughNets nmpFile1 feedthruNet_core*
  ```

insertPtnFeedBackBuffer

```
insertPtnFeedBackBuffer
     -bufCell cellName
     [-selectNet fileName]
     [-useShortName]
     [-allLowercase]
```

Inserts a feedthrough buffer to a net that loops back to an original partition to avoid the net routing over a partition area.

Parameters

`-allLowercase`	Specifies all lowercase characters for the inserted feedthrough buffer and net names.
`-bufCell` *`cellName`*	Specifies the name of the feedthrough buffer cell type.
`-selectNet` *`fileName`*	Specifies a file that contains net names to be feedthrough nets for a partition and feedthrough buffers inserted. If this parameter is not specified, feedthrough buffers are inserted for all nets that loop back to an original partition.
`-useShortName`	Specifies the abbreviation of inserted feedthrough net names so that the net names do not extend too long for multiple runs.

insertPtnFeedthrough

```
insertPtnFeedthrough
     [-selectNet fileName | -chanLess [-excludeNet fileName] |
          -ptnFile fileName]
     [-noBuffer | -bufCell {cellName}... [-doubleBuffer]]
     [-netMapping fileName]
     [-routeBased]
     [-useShortName]
     [-allLowercase]
     [-topoFile topologyFileName]
     [-saveTopoFile fileName]
     [-reduceAddedPort]
     [-preferPinAbutment]
     [-sameVoltage]
     [-excludePtnList {partitionName | partitionList}]
     [-selectMarkedNet]
     [-instPrefix instancePrefix]
     [-netPrefix netPrefix]
     [-checkOnly]
     [-verbose]
     [-ecoFile]
```

Inserts feedthrough buffers into the partitions, changing the original netlist, to avoid routing a net over a block area. ECO placement runs automatically with this command.

The `insertPtnFeedbackBuffer` command runs automatically after this command, for both placement-based and route-based feedthrough insertions. The feedback buffers are inserted using a placement-based algorithm for both place-based and route-based feedthrough insertions.

The buffer supplied during design import is taken as default. For Alternately, you can specify a buffer with the `-bufCell` parameter or specify the `-noBuffer` parameter to generate an `ASSIGN` statement in the netlist.

The `insertPtnFeedthrough` command can detect if the design has power domains. This way, the appropriate buffer is selected from the libraries tied to the power domain. For placement-based feedthrough insertion, buffers are inserted only for those power domains that have the power sequence set to *Always On*. For route-based feedthrough insertion, buffers are inserted based on the routing.

You can save the feedthrough buffer topology tree information in a file by using the `-saveTopoFile` parameter. You can later use the `-topoFile` parameter to replicate feedthrough buffer insertions for an ECO netlist by specifying the file that has the topology tree information. For more details of the flow, see Replicating Feedthrough Insertions Across ECO Netlists in the *Encounter User Guide*. You can also manually create a topology file to guide feedthrough insertion for specific nets. For more information, see Using a Topology File

to Insert Feedthrough Buffers section in the “Partitioning the Design” chapter of the *Encounter User Guide*.

Feedthroughs are inserted in nets that connect to floating ports. Floating ports are hierarchical ports that are not connected to any logic inside the module.

You can use this command after the floorplanning steps for a partitioned design.

The default behavior derives the feedthrough topology using a shortest-path algorithm, which is based on the placement results. The default behavior assumes a true channelless design—that is, the possibility of routing through channels is considered minimal. It does not consider routing congestion, routing blockages, route guides, pin guides, and bus guides. If these considerations are required, use the `-routeBased` parameter to derive the feedthrough topology based on the routing results.

Note: The `insertPtnFeedthrough` command removes nets that are inserted with feedthrough buffers from any net groups to which they belong. After running this command you should, therefore, update the net groups that contain feedthrough nets.

Note: If a pinguide exists for a net for which feedthrough buffers are inserted, the pinguide information is updated.

Parameters

`-allLowercase`	Specifies all lowercase characters for the inserted feedthrough buffer and feedthrough net names.
`-bufCell {`*`cellName...`*`}`	Specifies the name of the feedthrough buffer cell type. **Note:** You can specify multiple buffer cells with MSV designs. The first buffer will be used for the default power domain.
`-chanLess`	Specifies that the design is pure channelless, having no channel available for routing, and therefore all nets must be selected for feedthrough insertion. **Note:** This is the default.
`-checkOnly`	Specifies that the feedthrough buffers are *not* inserted. This parameter is useful when you want to know how the `insertPtnFeedthrough` command will affect the design. A summary information will be printed. You can specify this parameter along with the `-saveTopoFile` parameter. This will save the feedthrough topology information.

`-doubleBuffer` Adds a buffer close to the feedthrough input pin and adds a buffer close to the feedthrough output pin.

`-ecoFile` *`fileName`* Specifies the file for logging ECO directives. This parameter cannot be used with the `-checkOnly` parameter.

`-excludeNet` *`fileName`*

Specifies a file that contains net names to be excluded from being a feedthrough net for a partition.

Note: Use the `-excludeNet` parameter only with the `-chanLess` parameter.

`-excludePtnList {`*`partitionName`* `|` *`partitionList`*`}`

Specifies that feedthrough buffers should not be added to the specified partitions).

If you specify the parameter with the `-ptnFile` *`fileName`* parameter, all partitions in the file specified with the `-ptnFile` parameter will be considered for feedthrough insertion, even if the partitions are specified in with the `-excludePtnList {`*`partitionName`* `|` *`partitionList`*`}` parameter.

`-instPrefix` *`instancePrefix`*

Specifies a prefix for the newly created instances. The prefix that you specify will be used instead of the default prefix.

`-netMapping` *`fileName`*

Generates a file that maps the original net name to the new net name created for the feedthrough buffers.

you must specify this parameter if you later want to run the `hiliteFeedthroughNets` command to highlight the feedthrough paths of the nets.

`-netPrefix` *`netPrefix`*

Specifies a prefix for the newly created nets. The prefix that you specify will be used instead of the default prefix.

`-noBuffer` Generates an `ASSIGN` statement in the netlist instead of inserting a buffer.

`-preferPinAbutment`

Specifies that if a net logically belongs to only two abutted partitions, feedthrough buffers should not be inserted in the net.

For this parameter to be applicable, the following conditions should be met:

- The abutment between the two partitions should *not* be congested.
- Both partitions should have a rectangular boundary (rectilinear boundaries are not supported)
- Neither of the partitions should have multiple ports connnected to the net
- Neither of the partitions should have any floating port connected to the net

`-ptnFile` *`fileName`*

Specifies a file with the syntax to include or exclude partitions for a specified net. Based on the user-specified information, the Encounter software automatically derives the buffer tree topology.

`-reduceAddedPort`

Specifies that the feedthrough buffer insertion should follow the original routing topology more closely. This reduces the number of added ports and buffers.

Note: This option is available only for route-based feedthrough insertions.

`-routeBased`

Derives the feedthrough topology based on the routing results.

Note: Regardless of topology, for nets limited to two abutted partitions, feedthrough buffers are not inserted if the abutment is not congested.

`-sameVoltage`

For placement-based feedthrough insertion, specifies that feedthrough path should be specified through power domains that are at same voltage level.

Default: Feedthrough buffers are not specified through power domains that are at the same voltage levels.

Note: This parameter cannot be used with the `-routeBased` or the `-topoFile` parameters.

`-saveTopoFile` *`fileName`*

Saves a file with feedthrough buffer topology tree information from an existing floorplan.

`-selectMarkedNet` Specifies that feedthrough buffers should be inserted on the nets that are currently selected in the GUI.

`-selectNet` *`fileName`* Specifies a file that contains net names to be feedthrough nets for a partition and inserted feedthrough buffers.

`-topoFile` *`topologyFileName`*

Specifies the file that has the topology tree information. This information is used to create feedthrough buffers for the netlist.

If you use the `-topoFile` parameter, only those nets that are specified in the topology file are considered for feedthrough buffer insertion.

If a net does not exist in the design, it should not be in the topology file. For example, if ECO changes remove a net, that net should be removed from the topology file.

Note: Hierarchical bus names are not supported in the Encounter software. Therefore, you cannot use hierarchical bus names in the topology file.

Note: The names of the feedthrough nets, buffers, and ports created might change in the different runs of the command. Only the feedthrough path (the partitions through which the net will pass) will always remain the same across the different runs of the command.

`-useShortName` Specifies the abbreviation of inserted feedthrough net names so that the net names do not extend too long for multiple runs.

`-verbose` Specifies that net-specific information is included in the logs.

Default: Only the summary information is included in the logs.

Examples

- The following command generates a file, `ptn_file1`, with the syntax to include or exclude partitions for a specified net.

  ```
  insertPtnFeedthrough -ptnFile ptn_file1
  ```

  The created `ptn_file1` file contains the following:

  ```
  Net netname
  +Ptn ptnname1
  ```

```
+Ptn ptnname1
+Ptn ptnname1
EndNet
```

Where:

- `+Ptn` specifies a preferred partition for `insertPtnFeedthrough`.
- The specified partition is preferred, therefore it is not guaranteed to be picked up during feedthrough insertion.
- `-Ptn` excludes a partition during `insertPtnFeedthrough`. This prevents any buffer from being inserted into the specified partition.

- The following command generates a file, `topology_info`, with the feedthrough buffer topology tree information:

  ```
  insertPtnFeedthrough -saveTopofile topology_info
  ```

- The following command specifies that topology tree information from the `topology_info` file should be used to generate feedthrough buffer insertions:

  ```
  insertPtnFeedthrough -topofile topology_info
  ```

legalizePin

```
legalizePin
     [-ptn ptnName]
     [-pin {pinName | pinNameList}]
     [-moveFixedPin]
```

Moves a pin from its existing location in the following cases:

- The pin overlaps with any other object or pin. The objects considered are:
  - stripes
  - vias
  - feedthrough buffers
  - routing blockages
  - pin blockages
- The pin violates pin-to-pin spacing constraints
- The pin is on a non-preferred layer, a non-reserved layer, or the M1 layer
- The pin is not on the routing track
- The pin does not honor corner mask settings

The pin is assigned to the nearest legal location.

Note: If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.

Parameters

`-moveFixedPin`

If you use this parameter, the specified pins with a *Fixed* status can also be moved.

Note: Pins that are not specified but have a *Fixed* status are *not* moved.

Default: If you do not specify this option, the specified pins with a *Fixed* status are *not* moved.

`-ptn` *`ptnName`*

Specifies the partition for which the pins will be checked. This parameter is required if the partition is not the top-level module.

Note: The `legalizePin` command can also resolve I/O pins from overlapping power stripes. In this case, *`partitionName`* specifies the block-level design name.

`[-pin {`*`pinName`* `|` *`pinNameList`*`}]`

Specifies that the command should be run on the specified pin(s) only.

loadBlackBoxNetlist

`loadBlackBoxNetlist` *`netlist`*

Loads a netlist for a black box. You can use this command to incrementally load an updated netlist for a blackbox.

You can use this command before converting a blackbox to a fence.

Parameters

`netlist`	Specifies the netlist name. The netlist can contain more than one black box netlist, but it should not contain any non-blackbox netlist.

Example

The following command loads the netlist `netlist_v1`.

```
loadBlackBoxNetlist netlist_v1
```

loadPtnPin

```
loadPtnPin
     -ptnName partitionName
     {-infile floorplanFileName | -def defFileName}
     [-report reportFileName]
```

Loads either a partition floorplan file (from a saved partition design session) or a DEF file, and then displays the preassigned I/O pins for the partition. This command must be run for each partition in the design.

You can use this command on a partition design after loading the top level floorplan and after running/loading placement and trial route data. Use `unloadPtnPin` to remove the preassigned I/O pins.

Note: The `loadPtnPin` command can load partition files that are saved using the `saveFPlan` command.

Parameters

`{-infile` *`floorplanFileName`* `| -def` *`defFileName`*`}`

Specifies the full pathname of the partition floorplan file or the DEF file.

`-ptnName` *`partitionName`*

Specifies the name of the partition.

`-report` *`reportFileName`*

Specifies the optional report filename that contains any pins deleted or newly added.

partition

```
partition
     [-buildScan | -buildScanVerbose]
     [-noInheritPhysical]
     [-stripStayOnTop]
     [-pinCutSpace]
     [-noViaCutSpace]
     [-pushRoute]
     partitionName ...
```

Converts the specified module fences to be partitions, pushes down the physical cells or power routing information to the partition level design(s), and duplicates strips in the partition that overlap with the partition boundary at their original widths.

You can use this command after specifying the partition(s).

Note: To assign signal pins for the partitions, use the `assignPtnPin` command.

Note: This command should be used in combination with the `assignPtnPin` command. Cadence recommends not mixing these version 4.2 pin commands with previous (version 4.1 or earlier) pin commands in the design flow. See Assigning Pins in the “Partitioning the Design” chapter of the *Encounter User Guide* for more information.

Parameters

`-buildScan | -buildScanVerbose`
: Propagates scan chains to the partition blocks.

`-noInheritPhysical`
: Specifies that physical only (filler) cells are not pushed down into the partition; they remain at the top level.

`-noViaCutSpace`
: Specifies that there are no cuts in the obstructions around vias. You can use this parameter when you want to ensure that the top-level routing can connect the vias without using the space between the vias and the corresponding cover blockages.

`partitionName ...`
: Specifies the partition names to be committed.

`-pinCutSpace`
: Specifies that there are cuts in the obstructions around pins.

  Default: By default, there are no cuts in the obstructions around pins so that the top-level routing can connect the pins without using the space between the pins and the corresponding cover blockages.

`-pushRoute`	Pushes down signal routes when the design is partitioned. **Note:** For hierarchy-violating nets, the wire segments that have a hierarchy violation on the net are discarded. The other wire segments of hierarchy-violating nets are retained. For more information, see Pushing Down Signal Routes in the "Partitioning The Design" chapter of the *Encounter User Guide*
`-stripStayOnTop`	Specifies that stripes that are not on a layer reserved by the partition are retained at the top level and are also copied into the partition For more information, see the How Top-level Stripes Are Pushed Down in the "Partitioning The Design" chapter of the *Encounter User Guide*. **Note:** The `-stripStayOnTop` parameter is not applicable for X stripe pushdown.

Related Topics

- Perform Virtual Prototyping in the *Encounter Hierarchical Implementation Flow Guide*

pinAlignment

```
pinAlignment
     [-refObj refObjName]
     [-ptnInst ptnName]
     [-pinNames pinList]
     [-noSnap]
     [-keepLayer | -newLayer layer]
     [-legalizePin]
     [-markPlaced]
```

Aligns pins between blocks on their facing edges. By default, this command snaps pins to the routing grid.

Note: Specified pins will not be aligned if they do not have the reference connections on the reference block, or if the reference pins are not assigned prior to running this command.

Note: If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.

The pin alignment priority is as follows:

- If you specify the `-refObj` and the `-ptnInst` parameters but not the `-pinNames` parameter, the `pinAlignment` command aligns *all* pins of specified instances that are connected to the referenced object.
- If you specify the `-refObj` parameter but not the `-ptnInst` and the `-pinNames` parameter, the `pinAlignment` command aligns *all* pins of all partitions that are connected to the referenced object.
- If you specify the `-refObj`, `-ptnInst`, and the `-pinNames` parameters, the `pinAlignment` command tries to align *all* possible pins that can be aligned.
- If you do not specify the `-refObj` parameter, the following order of priority will be used for aligning partition pins that have multiple connections:
  - Hard macro pins
  - I/O pads or pins
  - Partition pins
  - Standard cell pins

Parameters

`-keepLayer | -newLayer` *`layer`*

Specifies whether to keep the current pin layer or specify a new pin layer.

Default: If you do not specify this option, the new pin layer will be the same as the referenced layer to reduce wire routing.

`-legalizePin` Specifies that the `legalizePin` command should run automatically after the `pinAlignment` command.

`-markPlaced` Specifies that the pins that are moved by this command will have a *Placed* status.

Default: By default, the pins moved by this command have a *Fixed* status.

`-noSnap` Disables pins from being snapped to the routing grid.

`-pinNames` *`pinList`*

Specifies the pin(s) to be aligned between the facing edges of the target block and reference block. You can use wildcards (`*?`) with this parameter.

If you specify more than one pin, you must enclose them in curly braces.

`-ptnInst` *`ptnName`* Specifies the target instance where its pins will be aligned to the pins of the specified instance.

`-refObj` *`refObjectName`*

Specifies the reference object where the specified pins of the target block will be aligned to the pins of the reference object.

The reference object can be a standard cell, an I/O pin, an I/O pad, or a macro.

Command Order

You can use this command after partitioning floorplanning.

Example

The following command aligns `A0` and `A1` pins of `p1` to the reference pins of `blockA`:

```
pinAlignment -refObj blockA -ptnInst p1 -pinNames {A0 A1}
```

pinAnalysis

```
pinAnalysis
     [-checkLegality]
     [-outFile fileName]
     [-noHtml]
```

Reports certain Quality of Results (QoR) metrics for pin assignment. The `pinAnalysis` command deletes the existing routes, reroutes the design ensuring that the routes pass through partition pins, and reports pin assignment QoR metrics.

Important

The `pinAnalysis` command creates new routes. The original routes are not retained.

Note: The `pinAnalysis` command reroutes the design using `trialRoute -honorPin`. This command thus takes at least as much time as running Trial Route.

In addition to displaying the report on screen, the command also generates the report in an HTML file named `pinAnalysis.html`.

For more information on the use-flow and metrics reported, see Reporting QoR of Pin Assignment Results in the *Partitioning the Design* chapter of the *Encounter User Guide*.

Parameters

`-checkLegality`	Checks the legality of pin assignment (similar to the `checkPinAssignment` command) and prints a summary report of the results. The legality details for every partition are available through hyperlinks that are provided in the HTML report file.
`-noHtml`	Specifies that HTML report file should not be created.
`-outFile` *`fileName`*	In addition to displaying the report on screen, prints the report to the specified file.

Command Order

You can run this command after assigning partition pins.

Example

The following command checks the legality of pin assignment, displays the QoR report for pin assignment on the screen, and also prints the QoR report to a file named `pinAnalysisMetricReport`.

```
pinAnalysis -checkLegality -outFile pinAnalysisMetricReport
```

ptnAwareRouteForPA

```
ptnAwareRouteForPA
     [trialRouteOptions]
     [-intraNets]
```

Provides a partition-aware routing topology, which is similar to the routing topology generated by `trialRoute -handlePartitionComplex`. However, the `ptnAwareRouteForPA` command generates this topology in much lesser time.

You can use this command after floorplanning the design and before assigning partition pins.

Parameters

`-intraNets`	Specifies that intra-partition nets should also be routed. *Default:* If you do not specify this option, only inter-partition and top-level nets are routed.
`trialRouteOptions`	Specifies the parameters to be provided to the `trialRoute` command that runs internally. For example you can specify the following: `-ignoreRoutingHalo -maxRouteLayer 5` **Note:** Do not specify the `-handlePartitionComplex` or the `-handlePartition` parameter; otherwise, an error will be generated and these parameters will be ignored.

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - Congestion-aware Pin Assignment for Channel-based Designs

Example

The following command creates a partition-aware routing topology that is similar to that created by `trialRoute -handlePartitionComplex`. The command invokes Trial Route with the `-keepExistingRoute` option.

```
ptnAwareRouteForPA -keepExistingRoute
```

pushdownBuffer

```
pushdownBuffer
     [-ptn partitionName]
     [-bufCell cellName]
     [-prefix instNamePrefix]
     [-useExistingPort {false | true}]
     [-instPrefix instPrefixName]
     [-netPrefix netPrefixName]
```

Pushes down instances and nets located within the fence of a partition, but that do not belong to the partition's netlist, into the design hierarchy. The physical routes belonging to the nets are deleted.

The instances are pushed down only if each instance cell satisfies the following criteria:

- The cell contains only one input term *and* only one output term.
- The cell has one, and only one, combinational timing arc.

  Note: The cell can have more than one timing arcs; however, there must be one, and only one, timing arc of the type combinational.

A prefix is added by default to the names of the instances or nets that are pushed down. To specify a custom prefix, use the `-instPrefix` or the `-netPrefix` parameter.

Note: The `-instPrefix` and the `-netPrefix` parameters apply only to instances and nets that are pushed down by the `pushDownBuffer` command. The new ports created by the command have the default prefixes, and the internal module nets are named based on the new ports.

Note: The instance(s) can be in a module that is not a parent or an ancestor of the partition module.

Parameters

`-bufCell` *`cellName`*	Specifies that only buffers that belong to type *`cellName`* are pushed down. *Default*: If you do not specify this option, all buffer cells in the partition are pushed down.
`-instPrefix`	Specifies a custom prefix for instances that are pushed down, instead of the default prefix.
`-netPreFix`	Specifies a custom prefix for nets that are pushed down, instead of the default prefix.

`-ptn` *`partitionName`* Specifies the partition.

Default: If you do not specify this option, the `pushdownBuffer` command operates on all partitions in the design.

`-prefix` *`instPrefixName`*

Specifies that only buffers that belong to *`instPrefixName`* are pushed down.

Default: If you do not specify this option, all instances in the partition are pushed down.

`-useExistingPort {false | true}`

This parameter is applicable for cases where the buffer being pushed down into the partition drives a port of the partition module.

- A value of `false` specifies that a new port should be created and the existing port should be detached.
- A value of `true` specifies that the existing port should be used and no new port should be created.

Default: The default value is `false`, that is, a new port is created and the existing port is detached.

Example

The following command pushes down instances, and assigns the prefix `PDI_` to the instances.

```
pushDownBuffer -instPrefix PDI_
```

recreatePtnCellBlockage

```
recreatePtnCellBlockage
     [partitionName]
     [-noPinCutSpace | -hasPinCutSpace]
     [-noViaCutSpace | -hasViaCutSpace]
```

Recreates cell blockage for a specified partition, or all partitions. This reflects incremental changes, such as pin movement.

You can use this command after editing the pin in a partition with the Pin Editor, or using the editPin command, to move pins in the partition.

Parameters

partitionName	Specifies the partition to recreate the cell blockage. *Default*: All partitions
-noPinCutSpace	Specifies that no cut space will be created for pins in the partition cell blockage. **Note:** Blockages are always cut for power pins.
-hasPinCutSpace	Specifies that cut space will be created for pins in the partition cell blockage. **Note:** This is the default.
-noViaCutSpace	Specifies that no cut space will be created for vias in the partition cell blockage.
-hasViaCutSpace	Specifies that cut space will be created for vias in the partition cell blockage. **Note:** This is the default.

reportIlmStatus

```
reportIlmStatus
```

Reports ILM Status information including ILM block names, SPEF files and timing constraint file names.

You can use this command at any time in the flow, but it only reports ILM information if you have already specified ILMs in the design.

Parameters

None

Examples

The following command generates the ILM status report for `tdsp_core`:

```
reportIlmStatus

*** ILM status report ***
Current design: tdsp_core
The following blocks are specified as an ILM block:
Leaf instance(Inst) DECODE_INST (cell: decode_i)
Leaf instance(Inst) DATA_BUS_MACH_INST (cell: data_bus_mach)
Current view is black-box or non-ILM (after unflattenIlm)
ILM block SPEF list       : ../ilm/decode_i/decode_i.spef
../ilm/data_bus_mach/data_bus_mach.spef
ILM timing constraint file: .constr.ilm.sdc
*** end of ILM status report ***
```

reportUnalignedNets

`reportUnalignedNets` *`fileName`*

Report the names of the nets where the pins are not aligned for the following cases:

- nets that have two pins on two different partitions and/or clones and are not connected to top level hard-macro, standard cell, or I/O.
- nets that have two pins that are connected to the same partition
- nets that have one pin connected to top level hard-macro, standard cell, or I/O and the other pin connected to a partition or a clone

Note: The nets reported by this command are also highlighted in the artwork window.

Note: The `assignPtnPin` command internally calls the `reportUnalignedNets` command to generate the summary report at the command line.

Parameters

`fileName`	Specifies an output report file.

The following are some of the attributes specified in the output report file to report the names of nets where the pins are not aligned:

- `Layer Mismatch`: Specifies nets that have pins at an aligned location but different layers.
- `Non-neighbor partition`: Specifies nets that have two pins on two different partitions that are not adjacent.
- `ptnToPtn`: Specifies nets that have two pins on two different partitions and/or clones.
- `topToPtn`: Specifies nets that have one pin connected to top level hard-macro, standard cell, or I/O and the other pin connected to a partition or a clone.
- `UnAligned`: Specifies nets that have unaligned pins.

Example

The following command creates an `a.out` report file:

```
reportUnalignedNets a.out
```

A summary of the reported unaligned nets displays on the Encounter console and in the Encounter log file. The following are some of the sample sections in the log file for the above command:

```
Unaligned topToPtn 2-pin Net: FE_FEEDX_NET_C__sort_rams__ram_sort_a_1___0_addr_1_.
Unaligned topToPtn 2-pin Net: FE_FEEDX_NET_C__eco_inst__sort_rams__0_dout_b0_0_.
Unaligned topToPtn 2-pin Net: FE_FEEDX_NET_C__eco_inst__sort_rams__0_dout_b0_1_.
...
Unaligned topToPtn 2-pin Net: ptr0[2].
Unaligned topToPtn 2-pin Net: ptr0[1].
Unaligned topToPtn 2-pin Net: ptr0[0].
..
There are 6 ptnToPtn 2-pin nets in the design.
There are 6 aligned ptnToPtn 2-pin nets.

There are 110 topToPtn 2-pin nets in the design.
There are 89 unaligned topToPtn 2-pin nets.
There are 14 layer mismatch topToPtn 2-pin nets.
There are 7 non-neighbour topToPtn 2-pin nets.

There are 116 total 2-pin nets in the design.
There are 89 unaligned total 2-pin nets.
There are 14 layer mismatch total 2-pin nets.
There are 7 non-neighbour total 2-pin nets.
There are 6 aligned total 2-pin nets.
```

The following is a sample of the resulting `a.out` report file

```
#################################################################
#  Generated by:      Cadence Encounter 09.10-a003_1
#  OS:                Linux x86_64(Host ID pe-opt6)
#  Generated on:      Wed May 20 10:30:39 2009
#  Command:           reportUnalignedNets a.out
#################################################################
FE_FEEDX_NET_C__sort_rams__ram_sort_a_1___0_addr_1_ | FE_FTN_181 (sort_rams) |
a[1] (ram_sort) | UnAligned | topToPtn
FE_FEEDX_NET_C__sort_rams__ram_sort_a_5___0_addr_5_ | FE_FTN_173 (sort_rams) |
a[5] (ram_sort) | UnAligned | topToPtn
FE_FEEDX_NET_C__eco_inst__sort_rams__0_dout_b0_0_ | dout_b0[0] (sort_rams) | Y
(eco_inst/FE_FTB_58) | UnAligned | topToPtn
...
FE_FEEDX_NET_C__eco_inst2__sort_rams__0_dataout_0_ | FE_FTN_109 (sort_rams) | Y
(eco_inst2/FE_FTB_31) | Layer Mismatch | topToPtn
```

```
FE_FEEDX_NET_C__eco_inst2__sort_rams__0_dataout_1_ | FE_FTN_104 (sort_rams) | Y
(eco_inst2/FE_FTB_29) | Layer Mismatch | topToPtn
FE_FEEDX_NET_C__eco_inst2__sort_rams__0_dataout_2_ | FE_FTN_99 (sort_rams) | Y
(eco_inst2/FE_FTB_5) | Layer Mismatch | topToPtn
...
dout[6] | dout[6] (sort_rams) | A (eco_inst2/FE_FTB_40) | UnAligned | topToPtn
dout[5] | dout[5] (sort_rams) | din[5] (ram_sort) | UnAligned | topToPtn
dout[4] | dout[4] (sort_rams) | A (eco_inst2/FE_FTB_41) | Layer Mismatch | topToPtn
...
ptr1[6] | ptr1[6] (sort_rams) | a[6] (ram_bank1) | UnAligned | topToPtn
ptr1[5] | ptr1[5] (sort_rams) | a[5] (ram_bank1) | UnAligned | topToPtn
ptr1[4] | ptr1[4] (sort_rams) | a[4] (ram_bank1) | UnAligned | topToPtn
...
ptr0[2] | ptr0[2] (sort_rams) | A (eco_inst/FE_FTB_65) | UnAligned | topToPtn
ptr0[1] | ptr0[1] (sort_rams) | A (eco_inst/FE_FTB_66) | UnAligned | topToPtn
ptr0[0] | ptr0[0] (sort_rams) | A (eco_inst/FE_FTB_67) | UnAligned | topToPtn
wr | wr (controller) | A (eco_inst1/FE_FTB_42) | Non-neighbor partition | topToPtn
```

resizeBlackBox

```
resizeBlackBox
     blackBoxName
     [-width width]
     [-height height]
     [-aspectRatio ratio]
```

Resizes a blackbox.

Parameters

`-aspectRatio` *`ratio`*	Specifies the new aspect ratio, which is height divided by width, with the blackbox area constant.
`blackBoxName`	Specifies the name of the blackbox.
`-height` *`height`*	Specifies the new height in user units
`-width` *`width`*	Specifies the new width in user units.

Examples

- The following command resizes the height of `blackbox1` to 600.5 µm. The width remains the same dimension.

  ```
  resizeBlackBox blackbox1 -height 600.5
  ```

- The following command resizes the height of `blackbox2` to be twice as long as its width. The area remains the same.

  ```
  resizeBlackBox blackbox2 -aspectRatio 2
  ```

saveBlackBox

```
saveBlackBox
     {instName | -all}
     [-5.3 | -5.4 | -5.5]
     [-lefFile fileName]
```

Creates a blackbox LEF file containing the pin definitions and obstructions.

Note: savePartition also automatically saves this file, among others.

Parameters

`[-5.3 | -5.4 | -5.5]`

Specifies the LEF file format version, either 5.3, 5.4, or 5.5.

Default: `5.5`

`-all` Specifies all blackbox instances.

`instName` Specifies the name of the blackbox instance. You can use wildcards (`*?`) with this parameter.

`-lefFile` *`fileName`* Specifies the LEF filename. If you do not specify a name, the default is *`cellTypeName`*`.lef`.

Example

The following command creates a LEF file containing the pin definitions and obstructions for all blackbox instances that start with the letter A to a LEF file named `BBox.lef`:

```
saveBlackBox A* -lefFile BBox.lef
```

savePartition

```
savePartition
     {[-noNetlist]
     [-noFPlan]
     [-def [-savePlacement]]
     [-pdef [-savePlacement]]
     [-tdf]
     [-defNoCutRow]
     [-dir dirName]
     [-pt]
     [-lib]
     [-snapBudget]
     [-pinLoad]
     [-driveCell | -inputTransition]
     [-inputLoad | -noInputLoad]
     [-ptnSite]
     [-rptNegSlackOnPorts value]
     [-topLevelNoPtnModule]
     [-scanDef]
     [-latencyOnClocks]
     [-noFalsePathsForUnCstrPorts]
     [-mergeClones]
     partitionName ... } |
     {[-oaPtnView oaPtnViewName]
     [-oaPtnLib oaPtnLibName]
```

Saves the partition information to the current directory or a specified directory. This command is run after running the `partition` command to generate timing budget files and stamp model files.

Note: When using the `savePartition` command, you cannot save the current floorplan to the partition directory so that it is available when restoring and flattening the design with the `loadPartition` command.

Parameters

`-def`	Writes out the DEF format files for the design.
`-defNoCutRow`	Specifies that, when row information is written to the DEF file, the rows are not cut against placement obstructions, block halos, or power domain fence minimum gaps.
`-dir` *dirName*	The directory name that will contain the partition information/ files.
`-driveCell`	Writes additional driving cell timing constraints.

`-inputLoad | -noInputLoad`

Specifies whether to write or not write out `set_load` constraints on input partition pins into the partition constraint file, which represents the load of the attached net.

Default: `-noInputLoad`

`-inputTransition` Writes out input transition time in the budgeting output files using the `set_input_transition` constraint.

`-latencyOnClocks` Writes the `set_clock_latency` constraint on the clocks instead of ports in the budgeting files.

Note: This parameter applies only to clocks in propagated mode.

Use this parameter when you have multiple clocks on the same port.

Default: Writes the `set_clock_latency` constraint on the ports

`-lib` Saves the library model.

Default: `-lib`

`-mergeClones` Save the partition data based on the worst-case derived budgets per-pin for the master/clones, with each pin on the master/clones reflecting the worst-case timing among the master/clones.

`-noFalsePathsForUnCstrPorts`

Specifies not to create false paths for unconstrained ports.

`-noFPlan` Saves partitions without generating a floorplan file for the partitioned design.

`-noNetlist` Saves partitions without generating a netlist file for the partitioned design.

`-oaPtnLib` *`oaPtnLibName`*

Specifies an OpenAccess database library directory where the top-level design and the block-level design for all the partitions will be saved.

This parameter is used when saving the partition into an OpenAccess database.

Note: If you want to save the partition data for each partition in a different view, run the `savePartition` command multiple times, specifying a different partition and view each time.

Note: The parameters specific to OpenAccess database cannot be specified with any other parameter.

`-oaPtnView` *`oaPtnViewName`*

Specifies the view name for the top and the partition views.

This parameter is used when saving the partition into an OpenAccess database.

Default: the default value of the view is `layout`

`partitionName ...` Specifies the names of each partition to be saved.

`-pdef` Writes out the PDEF format files for the design.

`-pinLoad` Writes additional pin load timing constraints.

`-pt` Specifies PrimeTime format.

Note: This is the default.

`-ptnSite` Outputs sites for partitions in the LEF file.

`-rptNegSlackOnPorts` *`value`*

Reports if partition ports have a slack less than the specified value. The slack is calculated based on data arrival time and clock time at data sampling points.

`-savePlacement` When used with the `-def` parameter, specifies that the *unplaced* standard cells should also be saved to the DEF file. This is equivalent to the `-unplaced` parameter of the `defOut` command.

When used with the `-pdef` parameter, specifies that the *placed* standard cells should also be saved to the PDEF file.

`-scanDef` Writes out a DEF file for each block with scan information.

`-snapBudget`	Assigns a percent of the external delay value to the module or instance name when the overall timing budget is exceeded. The external delay is reduced by the specified percentage in this case. If you do not specify this parameter, the software does not exceed the timing budget. You specify the percentage to be used for this parameter using the `setBudgetingMode` command.
`-tdf`	Writes out the TDF format files for the design.
`-topLevelNoPtnModule`	Suppresses stub verilog modules for partitions in the top-level partition netlist.

Example

- The following command saves the partition information/files in a directory `try3` hierarchically for each of the three partitions:

  ```
  savePartition -dir try3 TOPCHIP_SP sheet17 sheet25 sheet7
  ```

- The following command saves the partition information/files in the OpenAccess database format. The information for the top and the block level designs (all blocks) will be written in the `libForOA` directory view with the name `ptnView1`.

  ```
  savePartition -oaPtnLib libForOA -oaPtnView ptnView1
  ```

Related Topics

- Perform Virtual Prototyping in the *Encounter Hierarchical Implementation Flow Guide*

savePtnPin

```
savePtnPin
     {-ptn partitionName | -all | -design}
     fileName
```

Saves the pin assignment information for one or more partitions. You can subsequently load the saved pin assignment information with the `loadPtnPin` command. The `savePtnPin` command also saves the pin assignment information for blackboxes.

Parameters

`-all`	Specifies that pin assignment information will be saved for all partitions, including the top-level design.
`-design`	Specifies that pin assignment information will be saved only for the top-level design.
`fileName`	Specifies the name of the file in which the partition pin assignment information will be saved
`-ptn` *`partitionName`*	Specifies the name of the partition for which the pin assignment information is to be saved

Example

The following command saves the pin assignment information for the partition named `tdsp_block` into a file named `tdsp_block_pin_info`.

```
savePtnPin -ptn tdsp_block tdsp_block_pin_info
```

selectPtnPinGuide

```
selectPtnPinGuide
     llx lly urx ury
     pinGuideName
     metalLayer
     minSpace
     [pinGroupCell]
```

Selects a `partition` pin guide and highlights it in the design display window.

Parameters

llx	Specifies the lower left x coordinate of the pin guide.
lly	Specifies the lower left y coordinate of the pin guide.
metalLayer	Specifies the layer on which the pin guide resides.
minSpace	Specifies the minimum pin spacing for the pin guide.
pinGroupCell	Specifies the cell name if the pin guide is associated with a cell pin group.
pinGuideName	Specifies the name of the pin guide.
urx	Specifies the upper right x coordinate of the pin guide.
ury	Specifies the upper right y coordinate of the pin guide.

setActiveLogicViewMode

```
setActiveLogicViewMode
     [-help]
     [-reset]
     [-async {false | true}]
     [-highFanoutPort {true| false}]
     [-loopBack {false | true}]
```

Controls certain behaviors of active logic view commands.

Use the `setActiveLogicViewMode` command to display the current settings for the `connectMacroFeedthrough` command.

Parameters

`-async {false | true}`

Discards or retains the portion of the netlist relating to asynchronous control and scan controls terminals. By default, the tool discards this portion of the netlist. This option potentially conflicts the setting of `setAnalysisMode -async true` when the analysis for an asynchronous circuit is desired. When this happens, a warning message is issued.

Default: `false`

`-highFanoutPort {true | false}`

Retains or discards the portion of the netlist relating to high-fanout ports. By default, the tool retains this portion of the netlist.

Default: `true`

`-help`

Outputs a brief description that includes type and default information for each `setActiveLogicViewMode` parameter.

For a detailed description of the command and all of its parameters, use the man command:
`man setActiveLogicViewMode`

`-loopBack {false | true}`

Discards or retains the portion of the netlist relating to input and output loopback paths. By default, the tool discards this portion of the netlist.

Default: `false`

`-reset`

Resets the active logic view mode command parameters to their default values.

Example

The following command resets all the `setActiveLogicViewMode` parameters to their default values:

```
setActiveLogicViewMode -reset
```

setAllowedPinLayersOnEdge

```
setAllowedPinLayersOnEdge
     {-edge edgeName | -all}
     [-layer {layerId | layerIdList}]
     [-cell cellName]
```

Sets the pin layers for a specified edge or for all edges.

You can run this command during partition floorplanning.

Parameters

`-all`
: Specifies that the specified pin layer(s) should be set for all edges.

`-cell` *`cellName`*
: Specifies the name of the module.

  Default: if you do not specify this parameter, the command is run on the top cell.

`-edge` *`edgeName`*
: Specifies the name of the edge. This parameter can take the following values:

  - `N, S, W, E` (supports both upper and lower case): For North, South, West, or, East sides, respectively.
  - `T, B, L, R` (supports both upper and lower case): for Top, bottom, Left, or Right sides respectively.
  - `dbcN, dbcS, dbcE, dbcW`

`-layer` *`layerId | layerIdList`*
: Specifies the layer(s) that should be set for the specified edge.

Example

The following command sets pins layers `M5` and `M6` for all edges.

```
setAllowedPinLayersOnEdge -all -layer {5 7}
```

setClonePtnOrient

```
setClonePtnOrient
     [-help]
     instName
     orientation
```

Changes the orientation of the specified partition clone.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `setClonePtnOrient` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man setClonePtnOrient`.
`instName`	Specifies the name of the partition clone whose orientation has to be changed.
`orientation`	Specifies the orientation. The orientation can be one of the following: ■ `R0` ■ `MY` ■ `MX` ■ `R180` ■ `MX90` ■ `R90` ■ `R270` ■ `MY90`

Example

The following command sets the orientation of the partition clone name `TDSP_ARB_C1` to `R270:`

```
setClonePtnOrient TDSP_ARB_C1 R270
```

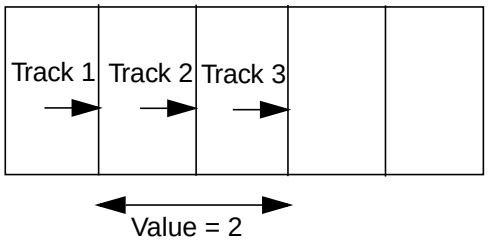
setGlobalMinPinSpacing

```
setGlobalMinPinSpacing
     minPinSpacing
```

Sets the global minimum pin spacing for partitions and blackboxes. You can use this command after importing the design.

Parameters

minPinSpacing	Specifies the pin spacing as the distance between tracks. For example, if you specify a value of 3, the global minimum pin spacing will be equal to the distance between track 1 and track 3.



Example

The following command sets the global minimum pin spacing to 5 tracks.

```
setGlobalMinPinSpacing 5
```

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - Setting Pin Constraints
  - Pin spacing

setIlmMode

```
setIlmMode
     [-help]
     [-reset]
     [-async {true | false}]
     [-highFanoutPort {true | false}]
     [-loopBack {true | false}]
```

Controls certain behaviors of ILM commands.

Use the getIlmMode command to display the current settings for the setIlmMode command.

Parameters

`-async {true | false}`

Keeps paths from asynchronous input ports when ILMs are created by createInterfaceLogic. For example, paths from an input port to the reset pin of interface sequential instances, but not to the internal sequential instances.

Default: `false`

`-help`

Outputs a brief description that includes type and default information for each setIlmMode parameter.

For a detailed description of the command and all of its parameters, use the man command: `man setIlmMode`.

`-highFanoutPort {true | false}`

Keeps high-fanout input port paths when ILMs are created by createInterfaceLogic.

Default: `true`

`-loopBack {true | false}`

Keeps loop back paths when ILMs are created by `createInterfaceLogic`.

When set to `true`, then any path touching a net used for the ILM model is also included. This is done for better model accuracy, so that the actual capacitance on an output port could affect the delay on a net which feeds an internal path that is normally not included in the ILM model. This also degrades the reduction ratio (`#ILM_insts/#all_inst`) for the block.

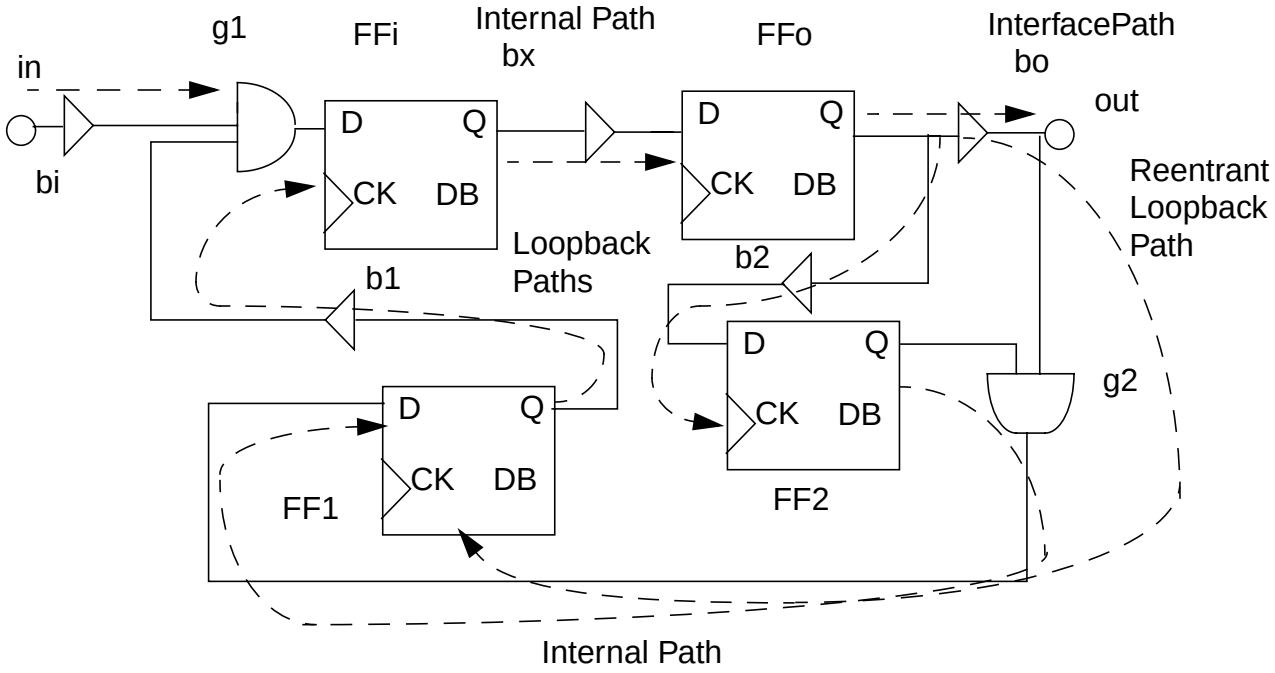
For more details, see `Loop-Back Paths`.

Default: `false`

`-reset`

Resets parameters to their default values. The `-reset` parameter must be the first parameter specified. If you specify reset by itself, the software resets all `setIlmMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

Loop-Back Paths



Examples

- The following command does not keep high fanout ports when ILMs are created by `createInterfaceLogic`:

  ```
  setIlmMode -highFanoutPort false
  ```

- The following command resets the `-highFanoutPort` parameter to its original value:

  ```
  setIlmMode -reset -highFanoutPort
  ```

- The following command resets all `setIlmMode` parameters to their default values:

  ```
  setIlmMode -reset
  ```

setIlmType

```
setIlmType
     [-model {timing | cts | si}]
```

Controls the behavior of commands in the presence of ILMs.

Use the getIlmType command to display the current settings for the setIlmType command.

Parameters

-model {timing | cts | si} Specifies the ILM model that is to be used by the flattenIlm command. All the timing-related commands require data to be in the flattened state (physical command require unflattened state).

Design commands such as optDesign, clockDesign, timeDesign, and so on automatically set this parameter before flattening ILMs. However, the CTE commands and some clock commands, which can be run in the flattened mode require manual settings.

- Use the cts model before running flattenIlm followed by the command which directly deals with the clocks.
- Use the si model for flattening the SI ILM model.
- Use the timing model for flattening the non-SI related timing commands.

Note: The unflattenIlm command changes the model setting to its default value.

Default: timing

Examples

- The following command is used to report clock tree in the presence of ILMs:

```
setIlmType -model cts
flattenIlm
```

```
reportClockTree
unflattenIlm
```

The `unflattenIlm` command resets `-model` to its default value `timing` since the next command is not a clock command.

- The following command can be used to get timing reports containing the SI push-out delays on nets using the `-setIlmType -model` command:

```
setIlmType -model si

# Flattens to the timing model

flattenIlm

# Reflattens to SI model, then does not unflatten (All other design
# commands unflatten upon exit, regardless of the flattened/unflattended
# state before invocation)

timeDesign -postroute -si

# Adds increemental delay column (for SI push-out delays) in timing output:

set_global_report_timing_format {instance arc cell fanout load slew delay
incr_delay arrival}

# Minimizes the width of the report such that it easily fits into the screen
# without wrapping

set_table_style -name report_timing -no_frame -indent 0

report_timing
```

Note: You can also invoke the Global Timing Debugger (Timing – Debug Timing – Generate)

setLayerPinDepth

```
setLayerPinDepth
     [-cell cellName]
     -layer {layerId | layerIdList}
     {-depth depth | -default | -minWidth}
```

Sets the pin depth for the specified pin layer(s).

You can use this command after importing the design.

Parameters

`-cell cellName`	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
`-default`	Specifies that the depth of pins should be set such that the minimum area rule for the layer is followed. If the pin width for a layer was set previously using the `setLayerPinWidth` command, the `setLayerPinDepth` command will use the specified width to calculate the depth that satisfies the minimum area. Otherwise, the command uses the minimum width of the layer as specified in the LEF file to calculate the depth that satisfies the minimum area.
`-depth depth`	Specifies the pin depth in microns.
`-layer {layerId | layerIdList}`	Specifies the layer(s) for which the pin depth is specified.
`-minWidth`	Specifies that the minimum width as specified in the LEF file should directly be used as the pin depth for that layer.

Example

The following command sets pin depth for layers `M3`, `M5`, and `M7` of module `ptn1` to 3 microns.

```
setLayerPinDepth -cell ptn1 -layer {3 5 7} -depth 3
```

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - Setting Pin Constraints

- Pin Size

setLayerPinWidth

```
setLayerPinWidth
     [-cell cellName]
     -layer {layerId | layeridList}
     {-width width | -default}
```

Sets the pin width, in microns, for the specified pin layer(s).

You can use this command after importing the design.

Parameters

`-cell cellName`	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
`-default`	Specifies that the default pin depth specified in the LEF file should be used.
`-width width`	Specifies the pin width in microns.
`-layer {layerId | layerIdList}`	Specifies the layer(s) for which the pin width is specified.

Example

The following command sets pin width for layers `M3`, `M5`, and `M7` of module `ptn1` to `1` micron.

```
setLayerPinWidth -cell ptn1 -layer {3 5 7} -depth 1
```

Related Topics

- “Partitioning the Design” chapter in the *Encounter User Guide*
  - Setting Pin Constraints
  - Pin Size

setMinPinSpacingOnEdge

```
setMinPinSpacingOnEdge
     [-cell cellName]
     {-edge edgeName | -all}
     -spacing minSpace
```

Sets the minimum pin spacing, for a specified module and edge(s). You can specify a top-level module with this command.

You can use this command after importing the design.

Parameters

`-all`	Specifies that the minimum pin spacing should be set for all edges.
`-cell cellName`	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
`-edge edgeName`	Specifies the name of the edge. This parameter can take the following values: ❑ `N, S, W, E` (supports both upper and lower case): For North, South, West, or, East sides, respectively. ❑ `T, B, L, R` (supports both upper and lower case): for Top, bottom, Left, or Right sides respectively. ❑ `dbcN, dbcS, dbcE, dbcW`
`-spacing minSpace`	Specifies the minimum spacing between the pins, in tracks.

Example

The following command sets the minimum pin spacing on all edges of the module `SH17` to 2 tracks.

```
setMinPinSpacingOnEdge -cell SH17 -all -spacing 2
```

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - Setting Pin Constraints

- Pin Spacing

setPinConstraint

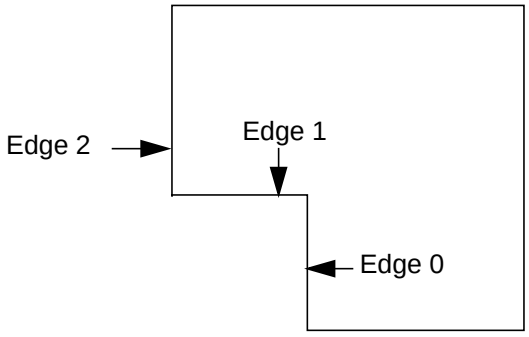
```
setPinConstraint
     [-cell cellName]
     -pin pinName
     {{[-loc x y z] | [-edge edgeNumber]}
     [-layer {layerId | layerIdList}]
     [-spacing minSpace]}
```

Sets the constraint for a partition pin or an I/O pin.

Parameters

`-cell` *`cellName`*
Specifies the name of the module. The module name is not required if the command is run on a top-level module.

`-edge` *`edgeNumber`*
Specifies the integer value where edge numbering starts from lower-left corner of a partition clock-wise. Edge `0` is the edge that has the smallest y value.



`-layer {`*`layerId | list_of_layers`*`}`

Specifies the metal layers for the partition pin.

If you specify multiple layers, separate the layer names with a space. In this case, the command will constrain the pin to one of the specified layers.

Note: This parameter does not restrict the pin to a location on the layer. If you want to restrict a pin on a particular location on a particular layer, use the `-loc` parameter instead.

Note: The `-layer` parameter cannot be used with the `-loc` parameter.

`-loc` *`x y z`*

Specifies the x and y locations and z layer number of the partition pin. The x and locations are calculated with respect to the distance in microns from the lower left corner of the module.

The layer number specified should be a preferred layer. Otherwise, the command will move the pin to a preferred layer.

For example, `-loc 10 12 7` specifies that the pin should be constrained to a location 10 microns on the x axis and 12 microns on the y axis from the lower left corner of the partition. on layer `M7`.

Note: If the pin location as per the specified x and y locations is not on a track, the pin is snapped to the nearest track on the specified layer.

Note: If you specify this parameter, the assigned pins have a *Placed* status.

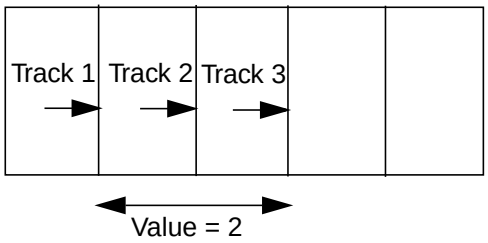
Note: The `-loc` parameter cannot be used with the `-layer` parameter.

`pinName`

Specifies the name of the partition pin. You can use wildcards (`*?`) with this parameter.

`-spacing min`*`Space`*

Specifies the minimum spacing between pins in pin pitch value, that is, the distance between tracks. For example, if you specify a value of 2, the minimum pin spacing will be equal to the distance between track 1 and track 3.



Command Order

You can use this command before assigning pins.

Example

The following command specifies a width of 10 micrometers for all partition pins in the partition `sheet14`. These pins will be placed either on layer 2 or on layer 4.

```
setPinConstraint -cell sheet14 -pin* -layer {2 4}
```

setPinToCornerDistance

```
setPinToCornerDistance
     [-cell cellName]
     {-corner cornerNumber | -all}
     pinDistance
```

Sets the pin-to-corner distance for a module. You can specify a top-level module with this command.

Note: If the shape of the partition or the blackbox is changed after you set the pin-to-corner distance with this command, all corners of the modified partition or blackbox will have the default pin-to-corner distance.

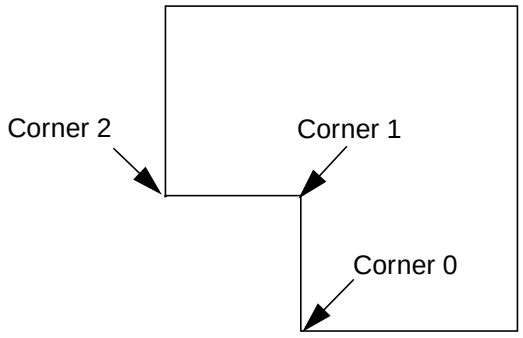
Parameters

`-all` Specifies that the pin-to-corner distance should set for all corners.

`-cell` *`cellName`* Specifies the name of the module. The module name is not required if the command is run on a top-level module. You can specify wildcards (?*) with this parameter.

`-corner` *`cornerNumber`*

Specifies the corner of the partition block. This is an integer value, where corner numbering starts at 0 from the lower-left corner of a partition clock-wise. Corner `0` is the corner that has the smallest y value.



`pinDistance` Specifies the pin-to-corner distance in tracks.

Command Order

You can specify this command during partition floorplanning.

Example

The following command sets the pin-to-corner distance for all corners of module `ptn1` to `10` tracks.

```
setPinToCornerDistance -cell ptn1 -all 10
```

setPinDepth

```
setPinDepth
     [-cell cellName]
     {-pin pinName | -pinGroup pinGroupName}
     {-depth depth | -default}
```

Specifies the depth for the specified pin or the pins in the specified pin group.

The pin depth set for a specific pin with the setPinDepth command will override any conflicting setting for the same pin specified with the setLayerPinDepth command.

You can use this command after specifying the partition—the one specified in the `-cell` *`cellName`* parameter.

Parameters

`cell` *`cellName`*	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
`-depth` *`depth`*	Specifies the pin depth in microns.
`default`	Specifies that the depth of the pin should be equal to the default value of the pin depth of the layer on which the pin exists.
`-pin` *`pinName`* `|` `-pinGroup` *`pinGroupName`*	Specifies the name of the pin or the pin group for which the pin depth is to be set.

Example

The following command sets the default pin depth for all pins in pin group `PG1` in module `PTN1` to 2 microns:

```
setPinDepth -cell PTN1 -pinGroup PG1 -depth 2
```

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - Setting Pin Constraints
  - Pin Size

setPtnPinStatus

```
setPtnPinStatus
     partitionName
     pinName
     {fixed | placed | unplaced | cover}
```

Sets the pin status for a pin of a partition as fixed, placed, or unplaced.

Note: Use this command to unassign blackbox pins *before* the blackbox is committed. To unassign blackbox pins *after* the blackbox is committed, use the `flattenPartition` command instead.

Note: If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.

Parameters

`cover`	Specifies a cover pin status.
`fixed`	Specifies a fixed pin status.
`partitionName`	Specifies the name of the partition. You can use wildcards (`*?`) with this parameter.
`pinName`	Specifies the name of the pin of the partition. You can use wildcards (`*?`) with this parameter.
`placed`	Specifies a placed pin status.
`unplaced`	Specifies an unplaced pin status.

Command Order

You can use this command any time after assigning pins.

setPtnPinUSE

```
setPtnPinUSE
     partitionName
     pinName
     {SIGNAL | CLOCK | ANALOG}
```

Allows you to change the + USE property of a partition pin.

Note: A partition pin can also be an I/O pin if the partition is a top-level cell.

Parameters

partitionName	Specifies the name of the partition.
pinName	Specifies the pin name.
SIGNAL | CLOCK | ANALOG	The choices represent + USE properties that you can change with this command.

Example

The following command sets the USE property of pin A in partition PTN1 to CLOCK:

```
setPtnPinUSE PTN1 A CLOCK
```

setPinWidth

```
setPinWidth
     [-cell cellName]
     {-pin pinName | -pinGroup pinGroupName}
     {-width width | -default}
```

Specifies the default pin width on a layer in the partition.

The pin width set for a specific pin with the `setPinWidth` command will override any conflicting setting for the same pin specified with the `setLayerPinWidth` command.

You can use this command during partition floorplanning.

Parameters

`cell` *`cellName`*	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
`-width` *`width`*	Specifies the pin width in microns
`default`	Specifies that the width of the pin should be equal to the default value of the pin width of the layer on which the pin exists.
`-pin` *`pinName`* `|` `-pinGroup` *`pinGroupName`*	Specifies the name of the pin or the pin group for which the pin width is to be set.

Example

The following command sets the default pin width for all pins in pin group `PG1` in module `PTN1` to 2 microns:

```
setPinwidth -cell PTN1 -pinGroup PG1 -width 2
```

Related Topics

- "Partitioning the Design" chapter in the *Encounter User Guide*
  - Setting Pin Constraints
  - Pin Size

showPtnWireX

```
showPtnWireX
     [ptnName]
     [-outfile fileName]
     [-excludeNet exNetFileName]
     [-excludeClock]
     [-excludeTri]
     [-excludeHighFan nrFan]
     [-allNets]
     [-delta]
```

Displays and writes a file of wires that physically cross over partitions.

You can use this command after routing the design.

Parameters

`-allNets`	Reports nets that cross partitions that also connect to the ports of partitions. If this parameter is not specified, only nets that cross the partitions are reported.
`-delta`	Reports the total segment length of the wires crossing the block in the X and Y directions.
`-excludeClock`	Excludes clock nets from the wire crossing information.
`-excludeHighFan nrFan`	Excludes nets with fanouts over *nrFan* from the display.
`-excludeNet exNetFileName`	Specifies the file name that contains nets to be excluded from the display.
`-excludeTri`	Excludes all tristate nets from the display.
`-outfile fileName`	Specifies the name of the output file of wire crossing information. *Default:* If you do not specify this parameter, it will print to the console.

ptnName	Specifies the name of the partition to show wire crossing. If you choose this option with other options, it must be the first specified when running this command. *Default:* If you do not specify this parameter, it will show all partition wire crossings.

Example

The following command displays the routes crossing over partition, `sheet17`, and outputs the detail information into file, `sheet17.crossing`:

```
showPtnWireX sheet17 -outfile sheet17.crossing
```

snapPtnPinsToTracks

```
snapPtnPinsToTracks
     ptnName
     [-snapOnLowerLayer]
     [-snapOnHigherLayer]
```

Snaps the center of partition pins to the nearest intersecting routing grid, where the horizontal and vertical routing tracks cross.

Note: If this command is run on a blackbox that has one or more pins with a single LEF port but with multiple shapes within the LEF port, the command will remove these multiple shapes and create a single shape. If this command is run on a blackbox that has one or more pins with multiple LEF ports, the first LEF port will be processed as per the command and the remaining LEF ports will remain unchanged.

You can use this command after running the Partition program.

Parameters

ptnName	Specifies the name of the partition. You can also specify I/O pins.
-snapOnHigherLayer	Specifies that the pin should be snapped to the cross point of the track on the layer of the pin and the layer immediately above the pin. For example, if the pin is on the M4 layer, the -snapOnHigherLayer specifies that the pin should be snapped to the cross point of the track on the M4 layer and the track on the M5 layer. *Default:* By default, this command snaps the pin to the intersection of the horizontal and vertical routing tracks on the layer on which the pin exists. For example, if a pin is on the layer M4, the command will snap the pin to the cross points of the tracks on the layer M4 itself.

`-snapOnLowerLayer`	Specifies that the pin should be snapped to the cross point of the track on the layer of the pin and the layer immediately below the pin. For example, if the pin is on the M4 layer, the `-snapOnLowerLayer` specifies that the pin should be snapped to the cross point of the track on the M4 layer and the track on the M3 layer. *Default:* By default, this command snaps the pin to the intersection of the horizontal and vertical routing tracks on the layer on which the pin exists. For example, if a pin is on the layer M4, the command will snap the pin to the cross points of the tracks on the layer M4 itself.

Example

The following command snaps center of partition `ptn_xy` pins to the nearest intersecting routing grid:

```
snapPtnPinsToTracks ptn_xy
```

specifyBlackBox

```
specifyBlackBox
     -cell cellName
     {-size x y
     | -area totalArea
     | -gateCount count areaPerGate utilizationValue
     | {-gateArea gateAreaValue [-cellUtil utilizationValue]
          [{-includeMacroArea {0 | 1}
          {-macroArea macroAreaValue
          | -macroList {macroName [count]}...}]}}
     [-minWidth x | -minHeight y | -fixedWidth x
          | -fixedHeight y] [-aspectRatio value]
          | [blackBoxName] instName
```

Converts a hard macro into a blackbox. It is also used for specifying an instance to be the master of multiplying instantiated blackboxes.

You can use this command after importing the design.

You can convert a hard macro into a blackbox; however, Cadence does not recommend this because it will not update the blockage definitions when you change the blackbox size.

Parameters

`-area totalArea`

Specifies the total area in square micrometers. If you specify the block size area, the default for the created block aspect ratio is 1.

`-aspectRatio value`

Specifies the aspect ratio value. This value combined with the area information determines the width and height.

`blackBoxName` Specifies the name of the blackbox.

`-cell cellName`

Specifies the name of the cell.

`-cellUtil utilizationValue`

Specifies the cell utilization. This value is used to derive the area of the blackbox, which is calculated using the following formula:

Blackbox area = (standard cell area / cell utilization) + macro area

Default: The default value of cell utilization is `1` (100%).

`-fixedHeight` *`y`*

Specifies the fixed height of the module.

`-fixedWidth` *`x`* Specifies the fixed width of the module.

`-gateArea` *`gateAreaValue`*

Specifies the gate area. If the value of the `-includeMacroArea` parameter is `0`, the gate area is equal to the standard cell area. If the value of the `-includeMacroArea` parameter is `1`, the gate area also includes the macro area.

`-gateCount` *`count areaPerGate utilizationValue`*

Specifies a gate count value. The total cell area is equal to the specified gate count multiplied by the specified area per gate.

`count`	Specifies the gate count value.
`areaPerGate`	Specifies an area per gate value in square micrometers. To get the cell area, this value is multiplied by the specified gate count.
`utilizationValue`	Specifies the cell utilization. This value derives the area that is reserved for routing. The final block size is equal to the total cell area plus the routing area. *Default*: The default value is 0.7 (70%)

`-includeMacroArea {0 | 1}`

A value of `0` specifies that the gate area does not include the macro area specified through the `-macroArea` or the `-macroList` parameter.

A value of `1` specifies that the gate area includes the macro area specified through the `-macroArea` or the `-macroList` parameters.

Default: The default value is `0`.

`instName` Specifies the name of an instance to be assigned as the blackbox master.

`-macroArea` *`macroAreaValue`*

Specifies the total area of the macro(s) in square microns.

- If the value of the `-includeMacroArea` parameter is set to 0:

  Total blackbox area = (gate area / cell utilization value) + macro area

- If the value of the `-includeMacroArea` parameter is set to 1:

  Total blackbox area = ((gate area - macro area) / cell utilization value) + macro area

`-macroList {`*`macroName`* `[`*`count`*`]}...`

Specifies the name(s) of the hard macro(s) whose area(s) will be included while calculating the total gate area. The `macroName` argument specifies the name of the macro, and the `count` argument specifies the number of times the macro is referenced. Thus, by specifying a value for count, you avoid repeating the macro names for macros that are referenced multiple times.

- If the value of the `-includeMacroArea` parameter is set to 0:

  Total blackbox area = (gate area / cell utilization value) + macro area

- If the value of the `-includeMacroArea` parameter is set to 1:

  Total blackbox area = ((gate area - macro area) / cell utilization value) + macro area

Default: The default value of *`count`* is `1`.

`-minHeight` *y* — Specifies the minimum height of the specified module.

`-minWidth` *x* — Specifies the minimum height of the specified module.

`-size x` *y* — Specifies fixed width and height of a specified module, in micrometers.

Examples

- The following command converts the hard macro `hn246` into a black box. The gate area is specified as 100000 square microns and the specified macro area is 220 square microns. The cell utilization value is not specified, and so the default value of 1 is used. The `-includeMacroArea` parameter is not specified and so the default value of 0 is used.

  When the `-includeMacroArea` parameter is 0, the black box area is calculated using the formula:

  Total blackbox area = (gate area / cell utilization value) + macro area

  The total black box area in this case is, therefore, (100000 / 1) + 220 that is 100220 square microns.

  ```
  specifyBlackBox -cell hn246 -gateArea 100000 -macroArea 220
  ```

- The following command is similar to the command in the previous example, except that the value of `-includeMacroArea parameter` is set to 1 and the value of the `-cellUtil` parameter is set to `.7`

  When the `-includeMacroArea` parameter is set to 1, the blackbox area is calculated using the formula

  Total blackbox area = ((gate area - macro area)/cell utilization value) + macro area

  The total blackbox area in this case is, therefore, ((100000 -220) /.7 + 220) that is 142762.86 square microns.

  ```
  specifyBlackBox -cell hn246 -gateArea 100000 -includeMacroArea 1 -macroArea 220
  -cellUtil 0.7
  ```

- The following command converts the macro `hn246` into a blackbox and specifies that the areas of macros `macro_2A` and `macro_2B` should be added while calculating the blackbox area. The macro named `macro_2A` is included four times and the macro named `macro_2B` is included three times.

  ```
  specifyBlackBox -cell hn246 -gateArea 100000 -macroList macro_2A 4 macro_2B 3
  ```

specifyIlm

```
specifyIlm
     -cell cellName
     -dir dirName
```

Specifies ILM data directory for the specified block. You can run `specifyIlm` multiple times in the same session. Each time you specify `specifyIlm`, the software overwrites the previous setting for the block. If master/clones exist in the design, the cell name will have the name of the master partition.

Note: You can use this command only if the ILMs are unflattened (`unflattenIlm`) or the design is in black box view. You cannot change ILM settings in flattened or ILM view.

The syntax is order-dependant. You must specify `-cell` before you specify `-dir`.

Parameters

`-cell` *`cellName`*	Specifies the block name.
`-dir` *`dirName`*	Specifies the directory for the files you want to read.

Examples

- The following commands specifies directory `dir_dsp_core` for block `tdsp_core`, and directory `dir_onemore` for block `onemore`:

  ```
  specifyIlm -cell tdsp_core -dir dir_dsp_core
  specifyIlm -cell onemore -dir dir_onemore
  ```

- The following commands show how to reset the ILM directory from `dir_tdsp_core` to `dir2_tdsp_core`:

  ```
  specifyIlm -cell tdsp_core -dir dir_tdsp_core
  specifyIlm -cell tdsp_core -dir dir2_tdsp_core
  ```

  These commands are equivalent to the following commands, which include `unspecifyIlm`:

  ```
  specifyIlm -cell tdsp_core -dir dir_tdsp_core
  unspecifyIlm -cell tdsp_core
  specifyIlm -cell tdsp_core -dir dir2_tdsp_core
  ```

specifyPartition

```
specifyPartition
     partitionSpecFileName
     [-noEqualizePtnHInst]
```

Loads the partition specification file. The partition specification file is created from the Specify Partition form.

You can use this command after importing the design.

Note: You can create partitions inside power domains as well.

Parameters

`-noEqualizePtnHInst` Disables the snapping capability of the clone partitions to the power grid, such that clones do not necessarily have the same row structure and pattern as their master.

`partitionSpecFileName`

Specifies the file containing the partition specification.

Example

The following command loads the partition specification file `app1.ptn` and updates the Floorplan view:

```
specifyPartition app1_p.ptn
```

unflattenIlm

```
unflattenIlm
```

Reverts the ILM back to its original hard macro view.

You can use this command after the `flattenIlm` command.

Parameters

None

Example

The following command flattens all partitions and their ILM designs to the top-level. After analysis, the design is reverted back with the `unflattenIlm` command.

```
flattenIlm
unflattenIlm
```

unloadPtnPin

```
unloadPtnPin
     -ptnName partitionName
     -infile floorplanFileName
```

Removes the loading of partition pins by the `loadPtnPin` command. This is used to remove the preassignment of pins that were previously reassigned by the loading.

Parameters

`-infile` *`floorplanFileName`*

Specifies the name of the partition floorplan file and must specify the path.

`-ptnName` *`partitionName`*

Specifies the name of the partition.

unsetPinConstraint

```
unsetPinConstraint
     [-cell cellName]
     -pin pinName
     {[-loc | -edge]
     [-layer]
     [-spacing]}
```

Removes the constraints for a partition pin. You can use this command after setting the pin's constraints with the `setPinConstraint` command.

Parameters

`-cell` *`cellName`*	Specifies the name of the module. The module name is not required if the command is run on a top-level module.
`-edge`	Specifies that edge-related constraints should be removed.
`-layer`	Specifies that the layer-related constraints should be removed.
`-loc`	Specifies that the location-related constraints should be removed.
`pin` *`pinName`*	Specifies the name of the partition pin. You can use wildcards (`*?`) with this parameter.
`-spacing`	Specifies that the spacing-related constraints should be removed.

Example

The following command removes the spacing- and layer-related constraints for all partition pins in the partition `sheet14`.

```
unsetPinConstraint sheet14 -pin * -layer -spacing
```

unspecifyBlackBox

```
unspecifyBlackBox
     {-cell cellName | -all}
     [-keepPtn]
```

Unspecifies a blackbox and converts it to a module if the blackbox was originally defined as a module, or a block if the blackbox has a cdump or LEF macro definition.

You can use this command after importing the design.

Note: You cannot unspecify a blackbox of an undefined module or cell.

Parameters

`-all`	Selects all blackboxes for unspecification.
`-cell cellname`	Selects an individual blackbox for unspecification. You can use wildcards (*?) with this parameter.
`-keepPtn`	Preserves the partition information. You should only use this option if the blackbox is placed inside the core area and is defined as a module in the netlist.

Example

The following command unspecifies all blackboxes that begin with the letter A—some of which are inside the core area, and some which are outside the core area—and converts them to either a fence or module.

```
unspecifyBlackBox -cell A*
```

unspecifyIlm

```
unspecifyIlm
     -cell cellName
```

Unspecifies the ILM cell.

Parameters

`-cell` *`cellName`*	The name of the block to unspecify.

updateBlock

```
updateBlock
     {{topDesignDir blockDesignDir...
          [{-topFP | -chipFP} FPFile]
          [-fe]} |
     {-topDesign {oaTopLib oaTopCell oaTopView}
          {-block {oaBlockLib oaBlockCell oaBlockView}}...}}
     [-all | {[-boundary] [-pin] [-placement]}]
     [-row]
```

Updates the specified block information back to the top-level design. You should run this command from the directory with the full chip-level floorplan.

You can use this command after implementing blocks.

Parameters

`-all` Updates the boundary, pin, and placement data.

`-block` *`oaBlockLib oaBlockCell oaBlockView`*

Specifies the block-level library, cell, and view names respectively for a design that was saved into an OpenAccess database.

This parameter is used when bringing back block data for the partition from an OpenAccess database.

You can specify multiple blocks with this parameter.

`blockDesignDir`

Specifies the path to block-level designs.

This design directory should be generated by the `saveDesign -def` command. If it is not, make sure that the Verilog netlist (`.v`) file and DEF (`.def`) file have the same name as its module name.

`-boundary` Updates the block boundary.

`-fe` Specifies that Encounter place-and-route data should be used.

Default: By default, data from DEF files is used.

`-row` Specifies that row information should be brought back from the block-level design.

`-pin` Updates the block pins.

`-placement` Updates the placement and blockages.

`topDesignDir`

Specifies the path to the top-level design created by the `saveDesign -def` command. This design directory should contain the Verilog netlist and DEF files.

`-topDesign {`*`oaTopLib oaTopCell oaTopView`*`}`

Specifies the top-level library, cell, and view names respectively for a design that was saved into an OpenAccess database.

This parameter is used when bringing back top-level data from an OpenAccess database.

`{-topFP | -chipFP}` *`FPFile`*

Specifies the chip-level floorplan file with partition information.

Note: The `-topFP` and the `-chipFP` parameters perform the same function. The `-topFP` parameter has been retained for backward compatibility.

Examples

- The following command updates the specified block information back to the top-level design. The top-level design data is in the `dirTop` directory and the block-level data is in the `dirBlock1` and `dirBlock2` directories. The boundary, pin, and placement data are also updated.

  ```
  updateBlock dirTop dirBlock1 dirBlock2 -all
  ```

- The following command updates the specified block information back to the top-level design. The design has been saved in an OpenAccess database and contains two blocks. The boundary, pin, and placement data are also updated.

  - The library, cell, and view names for the top-level are `libForOA`, `DTMF`, and `ptnView1` respectively.
  - The library, cell, and view names for the first block are `libForOA`, `TDSP_CORE`, and `ptnView1` respectively.
  - The library, cell, and view names for the second block are `libForOA`, `TDSP_ARB`, and `ptnView1` respectively.

  ```
  updateBlock -topDesign libForOA DTMF ptnView1 -block libForOA TDSP_CORE
  ptnView1 -block libForOA TDSP_ARB ptnView1 -all
  ```

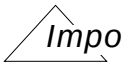
20

Placement Commands

addBlockFiller

```
addBlockFiller
     -cell cellName
     [-prefix prefix]
```

Adds filler cells to free block sites after placement, in designs that already have block sites in the floorplan. This command is used in structured ASIC designs.



Important

This command is not multiple supply voltage (MSV)-aware.

Parameters

`-cell` *`cellName`*	Specifies the cell master to add. The cell must be declared as `CLASS BLOCK` in the LEF file.
`-prefix` *`prefix`*	Specifies a prefix for the cell instance when it is added to the design. The new name for the cell instance is *`prefix_uniqueString`* *Default:* `FILLER`

Example

The following command adds filler cells of LEF `CLASS BLOCK` to `RAM_BLOCK` sites in a structured ASIC array. The cell instances have the prefix `RAMFILLER`.

```
addBlockFiller -cell RAM_FILLER_128x32 -prefix RAMFILLER
```

addDeCap

```
addDeCap
     -totCap capacitance
     [-cells cellName ...]
     [-area xl yl x2 y2]
     [-exclude {xl yl x2 y2} ...]
     [-effort [low | high]
     [-prefix prefixName]
     [-noFixDRC]
     [-force]
     [-log ecoFile]
     [-fromFile ecoFile]
     [-powerDomain powerDomainName]
     [-pwrNet netName]
```

Adds the specified total decoupling capacitance to the design. The software chooses from the available decoupling capacitance cell candidates, and adds enough cells until their combined total capacitance value equals the user-specified value.

Parameters

`-area` *`xl y2 x2 y2`*
Defines an area within the design in which to add the decoupling capacitance.
Default: Entire design area

x1	Specifies the area's lower left point on the x axis.
y1	Specifies the area's lower left point on the y axis.
x2	Specifies the area's upper right point on the x axis.
y2	Specifies the area's upper right point on the y axis.

`-cells` *`cellName`* …

Specifies the cells to use for decoupling capacitance insertion. The specified cells must first be defined using the `addDeCapCellCandidates` command.
Default: Any cell previously defined with the `addDeCapCellCandidates` command

`-effort [low | high]`

Specifies the approach to use when inserting decoupling capacitance.
Default: `low`

Specify one of the following:

`low`	Uses a homogeneous approach to add the decoupling capacitance. That is, the software treats all areas the same when adding decoupling capacitance cells.
`high`	Analyzes the peak current of the instances and tries to add more cells to the areas with high peak current density.

`-exclude {`*`xl yl x2 y2`*`} …`

Defines a blockage area within which decoupling capacitance cannot be inserted. You can specify one or more blockage areas.

`-noFixDRC` Disables correction of DRC errors.

`-force` Adds the decoupling capacitance at the specified location regardless of whether there are other instances located there. The software later calls the `refinePlace` command to move the instances.
Default: Searches for the closest legal location to add the decoupling capacitance in order to avoid conflict with the existing instances.

`-fromFile` *`ecoFile`* Performs decoupling capacitance insertion using the information in the specified log file (created from a previous `addDeCap` session).

This file can contain information for adding decoupling capacitance cells and for deleting existing decoupling capacitance cells.

`-log` *`ecoFile`*

Outputs a log file with the specified name that contains information on where the decoupling capacitance cells were added. It can also contain information on where cells were deleted due to DRC violations.

The log file can be used for tracking the location of decoupling capacitance and for debugging purposes. It can also be used to insert decoupling capacitance cells in a different session.

The log file is output in the following format:

```
ADD cellName locationX locationY [orientation] [instanceName]
DEL instanceName
```

It contains the cell name and the location where an instance of it was placed. It can also contain the orientation of the instance, and a name for the instance, if that name has not been used already.

For example, the following line shows that a new instance of `CELL_8` with the name `Instance100` was added at the location 100 200, with an orientation of R0:

```
ADD CELL_8 100 200 R0 Instance100
```

`-powerDomain` *`powerDomainName`*

Adds the decoupling capacitance to the specified power domain.
Default: Adds the decoupling capacitance to all of the power domains.

`-prefix` *`prefixName`* Defines the prefix to use for the decoupling capacitance cells.
Default: `DECAP`

`-pwrNet` *`netName`*

Adds decap cells on the specified power rail.
Default: Add decap cells irrespective of power nets.

`-totCap` *`capacitance`*

Specifies the total decoupling capacitance to add to the design, in femtofarads (fF). The software uses the specified decoupling cell candidates and adds cells until their combined capacitance value equals *`capacitance`*.

Command Order

Use this command after defining decoupling capacitance cell candidates with the `addDecapCellCandidates` command.

Example

The following command adds 1000 fF of capacitance to the design using DECAP1 and DECAP8 cells, in that order:

```
addDeCap -totCap 1000 -cells DECAP1 DECAP8
```

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - "Adding Decoupling Capacitance"

addDeCapCellCandidates

```
addDeCapCellCandidates
     {cellName capacitance | -fromFile fileName}
```

Defines the cells that can be used by the `addDeCap` command to insert decoupling capacitance into the design.

Parameters

`cellName capacitance`

Specifies the name and capacitance value in femtofarads (fF) for the cell candidate.

`-fromFile` *`fileName`* Defines the cells and capacitance values contained in the specified file as decoupling capacitance cell candidates.

Command Order

Use this command to define the candidate cells before using the `addDeCap` command.

Examples

The following commands define two decoupling capacitance cell candidates: `DECAP1` has a capacitance value of 10fF, and `DECAP8` has a capacitance value of 5fF.

```
addDeCapCellCandidates DECAP1 10
addDeCapCellCandidates DECAP8 5
```

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - "Adding Decoupling Capacitance"

addEndCap

```
addEndCap
     -preCap cellName
     -postCap cellName
     [-prefix prefixName]
     [-coreBoundaryOnly]
     [-flipY]
     [-powerDomain powerDomainName]
     [-area x1 y1 x2 y2]
```

Places physical-only end cap cells at the ends of the site rows. A single-height cap cell is required at the end of each row.

To delete the end cap cells, use the deleteFiller command.

Note: This command is not meant for the insertion of multiple-height cells.

Parameters

`-area x1 y1 x2 y2`

Specifies the coordinates of the bounding box, in microns. By default, the core box that encloses all core rows is used.

- *x1* Specifies the area's lower left point on the x axis.
- *y1* Specifies the area's lower left point on the y axis.
- *x2* Specifies the area's upper right point on the x axis.
- *y2* Specifies the area's upper right point on the y axis.

`-coreBoundaryOnly` Specifies that end-cap cells are placed only at the core boundary. This ensures that end-cap cells do not get placed at locations where site rows are obstructed by blockages.

`-flipY` Flips the orientation of the end-cap instances in Y direction if the site symmetry allows it.

`-postCap cellName` Specifies the name of the cell in the technology library to be used to cap the end of each site row.

`-powerDomain powerDomainName`

Specifies the power domain in which the end-cap cells are inserted. This identifies the area where to place the end-cap cells and connect to the correct power and ground nets.

`-preCap` *`cellName`* Specifies the name of the cell in the technology library to use to cap the beginning of each site row.

`-prefix` *`prefixName`* Specifies the prefix for the placed instances. The generated name for the instance uses this prefix as `prefix_`*`uniqueString`*.

Prefix Default: `ENDCAP`

Important

Do not use an underscore at the end of the prefix name.

Note: If the given postCap cell is the same as the preCap cell, and the cell and site symmetries allow it, the postCap instances will be flipped in the Y direction. For this, both cells must be provided on the same command line.

Examples

- The following command preplaces cell `abc` at the beginning of each site row of the design, and cell `xyz` at the end of each site row of the design in `R0` orientation, and adds the prefix `PwrCap` for the placed instances.

  ```
  addEndCap -prefix PwrCap -preCap abc -postCap xyz
  ```

- The following command deletes all physical-only cells with the prefix `PwrCap`. As a result, all cells created in the previous example are deleted from the design.

  ```
  deleteFiller -prefix PwrCap
  ```

- The following command preplaces cell `abc` at the beginning and end of each site row of the design and, if site symmetry allows, flips the post-cap cell in Y direction. Thus, for N-oriented rows, placed pre-cap cell is in N orientation, and post-cap cell is in FN orientation.

  ```
  addEndCap -prefix PwrCap -preCap abc -postCap abc
  ```

- The following command, in which the pre-cap cell is the same as the post-cap cell, places pre-cap cells in FN orientation and post-cap cells in N orientation in N-oriented rows, if site symmetry allows:

  ```
  addEndCap -prefix PwrCap -preCap abc -postCap abc –flipY
  ```

- The following command flips both the pre-cap and post-cap cell in Y direction. This preplaces both cells `abc` and `xyz` in FN orientation in N-oriented rows:

  ```
  addEndCap –prefix PwrCap –preCap abc –postCap xyz -flipY
  ```

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - "Adding End-Cap Cells"

addFiller

```
addFiller
     [-help]
     -cell {filler_cell_list}
     [-area x1 y1 x2 y2]
     [-createRows {true | false}]
     [-doDRC {true | false}]
     [-ecoMode {true | false}]
     [-fitGap]
     [-fixDRC]
     [-honorPrerouteAsObs {true | false}]
     [-markFixed]
     [-minHoleCheck {true | false}]
     [-merge {true | false}]
     [-powerDomain powerDomainName]
     [-prefix prefixName]
     [-util targetUtilization]
```

Inserts filler cell instances in the gaps between standard cell instances. If the design is routed, this command also does DRC checks of the filler cells added to the wires in the design. It does not check adjacent cells. This process ensures that this command is fast enough to be used many times in the design flow.

> **Note:** Before using `addFiller`, run the globalNetConnect command to provide global-net-connection rules for supply pins of the added fillers. Without these rules, the built-in design-rule checks of `addFiller` will not be accurate.

To resolve adjacent cell DRC violations created by the insertion of filler cells, run the `verifyGeometry` command to mark these violations. Then, use the `addFiller` command again with the `-fixDRC` parameter to replace the filler cells that caused the violations.

> **Note:** This command resolves only filler-related violations—it cannot resolve net-based violations. To repair net-based violations, reroute the net with violations.

The `addFiller` command recognizes up to five implant layers and attempts to honor implant layer width and spacing rules within each implant layer. If it does not find the correct filler cell to add to an implant layer, it tries to add “no-implant” filler cells. If it cannot add the “no-implant” cells, it inserts a filler cell that is large enough that it does not violate the minimum width rule of the relevant implant layer.

Use this command before doing detailed routing. Use the deleteFiller command to remove added filler cell instances.

Important

Some `addFiller` parameters can also be specified with `setFillerMode`. However, the parameters you specify with `addFiller` apply to the current `addFiller` run only, and supersede parameters set by `setFillerMode`. For more information, see `setFillerMode`.

Parameters

`-area` *`x1 y1 x2 y2`*

Adds filler cells to the specified area. If the area is not defined properly, does not add filler cells.

Normally, designs are placed before filler cells are added. However, If you use this parameter and the design is not placed, you get a warning, and the command adds fillers in the specified area.

Note: You can specify this parameter with `addFiller` only, and not with `setFillerMode`.

`x1` Specifies the area's lower left point on the x axis.

`y1` Specifies the area's lower left point on the y axis.

`x2` Specifies the area's upper right point on the x axis.

`y2` Specifies the area's upper right point on the y axis.

`-cell` *`filler_cell_list`*

Specifies the list of filler cells to add. Separate filler cell names with spaces and enclose the list with quotation marks (") or curly braces (`{}`).

Note: Cells specified by this parameter override the list of cells provided by the `-core` parameter of `setFillerMode`.

`-createRows {true | false}`

Controls whether `addFiller` creates rows in the floorplan if a cell's site does not have rows. Specify `true` for this parameter to prevent `addFiller` from creating the rows.
Default: `true`

`-doDRC {true | false}`

Controls postroute DRC violation checking against existing net routing.

This parameter does not disable regular placement checks.

Default: `true`. If filler cells are added after routing, the software checks for DRC violations between regular nets and the filler cells.

`-ecoMode {true | false}`

Controls whether the software resolves overlaps of standard cells with fillers by removing the overlapping fillers, and inserting new fillers in any existing or newly created gaps due to partial overlaps. If this parameter is used after routing, the command also resolves DRC violations of filler geometries of all fillers with regular wires.
Default: `false`

`-fitGap`

Fills a gap between cells by adding a combination of cells, instead of by adding the single largest cell that fits, if doing so avoids leaving an unfilled single-width gap.

`-fixDRC`

Corrects DRC violations reported by `verifyGeometry` between filler cells and adjacent standard cells.

Note: The `addFiller` command resolves only filler-related violations—it cannot resolve net-based violations. To repair net-based violations, reroute the net with violations.

Use this parameter in subsequent runs of the `addFiller` command, after adding filler cells and checking for violations with `verifyGeometry`. The command replaces filler cells that cause DRC violations with fillers that do not cause violations. If it cannot replace a violating filler cell without causing another violation, it leaves a gap.

Note: You can specify this parameter with `addFiller` only, and not with `setFillerMode`.

`-help`

Outputs a brief description that includes type and default information for each `addFillerGap` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man addFillerGap`.

`-honorPrerouteAsObs {true | false}`

Instructs `addFiller` to honor the setting of the setPrerouteAsObs command. That is, when this parameter is `true`, `addFiller` treats preroutes on metal layers where filler cells have geometries as obstructions, and ignores preroutes on other layers. In most cases, this means that `addFiller` treats preroutes on *metal1* as obstructions and ignores preroutes on other metal layers. For example, if `setPrerouteAsObs` is set to *metal1* and *metal2*, and all the filler cell geometries are on *metal1*, `addFiller` treats *metal1* preroutes as obstructions but not *metal2* preroutes.
Default: `false`. `addFiller` ignores `setPrerouteAsObs` and inserts fillers in all legal sites.

`-markFixed`

Places newly added filler cells in `FIXED` status in the Encounter database.

Note: You can specify this parameter with `addFiller` only, and not with `setFillerMode`.

`-merge {true | false}`

When checking for spacing violations, merges same-net geometries to determine width-dependent spacing. If the LEF file contains width-dependent rules, this parameter causes `addFiller` to have the following behavior:

- If you specify `true`, the command merges overlapping same-net objects to create a large rectangle.
- If you specify `false`, the command uses the shape of the object from the database.

Default: `false`

Tip

To avoid spacing violations, Cadence recommends that you run `addFiller` or `setFillerMode` with the `-merge` set to `true` if you run `verifyGeometry` with the `-merge` parameter specified.

`-minHoleCheck {true | false}`

Calls `verifyGeometry` to check for minimum hole violations before adding filler cells and repairs the violations when `addFiller` runs.
Default: `false`

`-powerDomain` *`powerDomainName`*

Specifies the power domain in which to add filler cells.
Default: If this parameter is not specified, and a list of filler cells is specified with the `-cell` parameter, the `addFiller` command tries to add fillers to all power domains.

Note: You can specify this parameter with `addFiller` only, and not with `setFillerMode`.

`-prefix` *`prefixName`* Specifies the prefix for the placed instances.
Default: `FILLER`

`-util` *`targetUtilization`*

Specifies the percent of free space remaining after placement to fill with given filler cell instances.

Use `-util` to add gate-array core filler cells to use later in the design flow, along with gate-array core cells, for metal-only mask changes.

Note: You can specify this parameter with `addFiller` only, and not with `setFillerMode`.

Range: More than 0.0 (0%) and equal to or less than 1.0 (100%)

Default: `1.0`

Examples

The following command inserts filler cell instances for cells `FILL64`, `FILL16`, `FILL4`, `FILL2`, and `FILL1`:

```
addFiller -cell FILL64 FILL16 FILL4 FILL2 FILL1 -prefix FILLER
```

The following commands show the use of the `-util` parameter to set the utilization of GACORE filler cells to 50 percent:

```
# Place the design
placeDesign
# Now fill with GACORE cells up to 50% of remaining space
```

```
# area utilization
addFiller -cell {gcore16 gcore8 gcore4} -prefix GACORE -util 0.5
# Now fill with the standard filler cells
addFiller -cell {fill8 fill4 fill2 fill} -prefix FILLER
# Continue with routing
```

Related Topics

- ECO Flows chapter in the *Encounter User Guide*
  - Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
  - Route the Design and Run Postroute Optimization for Partitions in the *Encounter Hierarchical Implementation Flow Guide*
  - Route the Design and Run Postroute Optimization for Top-Level in the *Encounter Hierarchical Implementation Flow Guide*

addFillerGap

`addFillerGap` *`min_gap_in_microns`*

Moves placed standard cell instances within a row to create gaps so that filler cells can be added. Tries to move cells to create a gap of the specified size—if enough space is not available, moves cells until they abut each other. If a gap already exists, and it is larger than the specified size, the command does not move cells to make it smaller.

By default, this command deletes all routing, because the routing is no longer valid due to the movement of the cells. To override this behavior, specify the following command:

```
setPlaceMode -preserveRouting
```

This command has the following limitations:

- It does not moved `FIXED` cells.
- It does not move cells to other rows.
- It does not guarantee to that all gaps will be larger than the specified minimum gap size.

Use this command after placement and before routing.

Parameters

`min_gap_in_microns`

Specifies the minimum-size gap in which a filler call can be added.

Example

The following command moves standard cells within a row to create gaps of at least two microns between the cells:

```
addFillerGap 2
```

addSpareInstance

```
addSpareInstance
     [-help]
     -file fileName
     [-prefix prefix]
     [-clock netName]
     [-tie {0 | 1}]
```

Specifies a file that lists spare cells to add to the netlist. You can use this command before or after standard cell placement.

- If you use this command before placement, the `placeDesign` command distributes and places the added instances without including them in spare modules. It adds the instances to the top level of the design by default.
- If you use this command after placement, the software places the added instances without creating spare modules or disturbing the existing placement. The distribution of the cells is dependent upon the number of instantiations of each cell type.

Parameters

`-clock` *`netName`*	Specifies the clock net that is tied to the clock pins.
`-file` *`fileName`*	Specifies a text file that lists the spare cells, and the number of each, to add to the netlist. The file has the following format: "*`cellName number`*" Begin each line of a comment with a number sign (#).
`-help`	Outputs a brief description that includes type and default information for each `addSpareInstance` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man addSpareInstance.`
`-prefix` *`prefix`*	Specifies a prefix for the names of the newly added cells. *Default:* `FE_spare`
`-tie {0 | 1}`	Specifies the logical net to which the cells are connected. *Default:* `1`

Example

The following command adds instances of the cells specified in `spare_file` and ties them to logical net `0`. It ties the clock pins to `sysclk`:

```
addSpareInstance -file spare_file -tie 0 -clock sysclk
```

`spare_file` contains the following:

```
DFFNSRX2T1T10   1
INVX4T1T10      2
NOR2X2T1T10      2
```

addTieHiLo

```
addTieHiLo
     [-cell "tieHighCellName tieLowCellName" | -cell tieHighLowCellName]
     [-createHierPort {true | false}]
     [-postMask {true | false}]
     [-powerDomain powerDomainName]
     [-prefix prefixName]
     [-reportHierPort {true | false}]
     [-matchingPDs {true | false}]
```

Adds instances of specified tie-off cells to the logical hierarchy of the design and connects the tie-off pins of netlist instances to the tie-off pins of the added instances. Thus, these instances provide the connectivity between the tie-high and tie-low pins of the netlist instances to power and ground.

Note: If you do not specify parameters, this command uses the global parameters specified by the `setTieHiLoMode` command.

Parameters

`-cell cellNames`
: Specifies the tie cells to add.

  You can only specify a maximum of two tie-cells, where one cell must be a tie-high driver, and the other a tie-low driver. The two tie-cell names must be surrounded by quotation marks.

  This parameter overrides cell name(s) specified by the `setTieHiLoMode` command.

`-createHierPort {true | false}`
: Allows the added tie cells to connect to tie pins across hierarchical boundaries if other constraints, such as maximum fanout and maximum distance, allow it.
  Default: `false`

`-postMask {true | false}`
: Instructs the software to reuse existing tie cells to tie off a newly created spare instance in the design instead of adding or deleting tie cells. Searches for input pins that are tied to logical `1` or logical `0` and attempts to them off to tie high or tie low cells.

`-powerDomain` *`powerDomainName`*

Specifies a power domain in which to insert and place the tie-cells and connect to the correct power and ground nets.
Default: all power domains

`-prefix` *`prefixName`* Specifies a prefix for the added tie-cell instances.

`-reportHierPort {true | false}`

When the value of `-createHierPort` is `true`, reports created port connections to a file named `tiehilo.rpt`.
Default: `false`

`-matchingPDs { true | false}`

Adds tie-cells for all the power domains whose library bindings contain the specified tie-cells, if the power domain is not specified in the command line.
Default: `false`

Examples

- The following command adds tie-high and tie-low connectivity for designs with libraries that have one cell that provides both tie-high and tie-low connections:

  ```
  addTieHiLo -cell TIEOFF -prefix tieOff
  ```

- The following command adds tie-high and tie-low connectivity for designs with libraries that have separate tie-high and tie-low cells:

  ```
  addTieHiLo -cell "TIEHI TIELO"
  ```

- The following commands add tie-off connectivity for MSV designs with two power domains named `PD1` and `PD2`. The last command is necessary for the part of the design that is not in either of the two specified power domains.

  ```
  addTieHiLo -cell "TIEHI TIELO" -powerDomain PD1
  addTieHiLo -cell "TIEHI TIELO" -powerDomain PD2
  addTieHiLo -cell "TIEHI TIELO"
  ```

Related Topics

- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*

- Run Top-Level Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*

addWellTap

```
addWellTap
     -cell cellName
     {-maxGap microns | -cellInterval microns}
     [-area x1 y1 x2 y2]
     [-checkerBoard]
     [-fixedGap [-pitch microns [-pitchOffset microns] ] ]
     [-incremental list_of_cells]
     [-inRowOffset microns]
     [-startRowNum number]
     [-powerDomain powerDomainName]
     [-prefix prefixName]
     [-skipRow number]
```

Places physical-only well-tap cells.

- To delete well-tap cells, use the deleteFiller command.
- To verify well-tap placement, generate markers for violations caused by well-tap cells, and create a violation report, use the verifyWellTap command.

Parameters

`-area x1 y1 x2 y2`

Specifies the coordinates of the bounding box, in microns. By default, the core box that encloses all core rows is used.

x1 Specifies the area's lower left point on the x axis.

y1 Specifies the area's lower left point on the y axis.

x2 Specifies the area's upper right point on the x axis.

y2 Specifies the area's upper right point on the y axis.

`-cell cellName`

Specifies the name of the cell in the technology library to use as a well-tap. The cell should be of class type CORE.

`-cellInterval microns`

Specifies the maximum distance from the center of one well-tap cell to the center of the following well-tap cell in the same row. If well-tap cells cannot be placed at this location due to a placement blockage, the software searches for a legal location: first, towards the previous well-tap cell, and then farther away.

`-checkerBoard`

Places the well-tap cells in a checkered pattern, that is, with every other row offset by one-half gap. In rows where a one-half offset gap is not available in the row above or below, the tap-cell is inserted at one-half gap in the row in question.Thus, for a simple floorplan, the first and last rows will have well-taps at one-half gap only, while all other rows will have the full gap.

When a floorplan has hard blocks or placement blockages, the rows above and below the block (or blockage) will also have well-taps at one-half gap.

Note: When this parameter is specified, `-skipRow` is ignored.

`-fixedGap`

Ensures that the specified gap is always maintained. That is, the software does not search for a legal location if it cannot place well-tap cells at the specified `-maxGap` or `-cellInterval`. Instead, it goes on to place well-tap cells at the next position.

Tip

You can modify the gap by using the `-pitch` and `-pitchOffset` parameters.

`-incremental` *`list_of_cells`*

Checks whether any instances of the cells specified in the *`list_of_cells`* are within 0.45 times the `-maxGap` or `-cellGap` from the next well-tap cell in the same row. If so, does not add well-tap cells at that location.

`-inRowOffset` *`microns`*

Specifies the offset, in microns, of the first well-tap cell from the start of the site row.
Default: `0` (first cell on beginning of row is the starting point).

`-maxGap` *`microns`*

Specifies the maximum distance from the right edge of one well-tap cell to the left edge of the following well-tap cell in the same row. If well-tap cells cannot be placed at this location due to a placement blockage, the software searches for a legal location: first, towards the previous well-tap cell, and then farther away.

`-pitch` *`microns`*

Used with the `-fixedGap` parameter to give the software added flexibility to place well-tap cells if it is unable to place a cell at the specified location. If you specify this parameter, the software first tries to fit the well-tap cells in the locations specified by `-cellInterval` or `-maxGap`, or forced by `-fixedGap`; if it cannot do so, it adds well tap cells at the closest location allowed by this parameter.

`-pitchOffset` *`microns`*

Used with the `-fixedGap` and `-pitch` parameters to give the software greater flexibility. If you specify this parameter, the software first tries to fit the well-tap cells in the locations specified by `-cellInterval` or `-maxGap`, or forced by `-fixedGap`; if it cannot do so, it adds well tap cells at the closest location allowed by this parameter.

`-powerDomain` *`powerDomainName`*

Specifies the power domain in which the well-tap cells are inserted. This is the area where the well-tap cells are placed and connected to the correct power and ground nets.

`-prefix` *`prefixName`*

Specifies the prefix for the placed well-tap cells. The software adds an underscore and a unique string to the prefix, in the following format: `prefix_`*`uniqueString`*.

Prefix Default: `WELLTAP`

Important

Do not include an underscore when you specify a prefix.

`-skipRow` *`number`*

Specifies the number of rows to skip between two rows of an identical pattern.

Note: If you specify `-checkerBoard`, this parameter is ignored.

Default: `0` (No rows are skipped, so well-tap cells are added in the same pattern to every row)

`-startRowNum` *`number`*

Specifies the row number starting from the from bottom or left side of the design on which the well-tap cells must be placed.

Default: `1` (first row from the bottom is the starting point)

Examples

- The following command places instances of well-tap cell `xyz` on every row in the design. Instances in the same row are at least 22.50 μm but no more than 50 μm apart. Each instance name is prefixed with `welltap_`.

  ```
  addWellTap -cell xyz -prefix welltap -cellInterval 50
  ```

- The following commands place instances of well-tap cell `xyz` in a checkered pattern. The first instance in the odd row numbers (`1`, `3`, `5`, and so on) is placed at `0` (zero) offset. The first instance in even row numbers (2, 4, 6, and so on) is placed at 50 μm offset. Instances in the same row are at least 45 μm but no more than 100 μm apart. Instances on the odd rows are named `wtap_odd_1`, `wtap_odd_2`, and so on. Instances on the even rows are named `wtap_even_1`, `wtap_even_2`, and so on.

  ```
  addWellTap -cell xyz -prefix wtap_odd -cellInterval 100 -skipRow 1
  addWellTap -cell xyz -prefix wtap_even -cellInterval 100 -skipRow 1 \
      -startRowNum 2 -inRowOffset 50
  ```

- The following command places instances of well-tap cell `WTAP` on every row in the design. Instances in the same row are at least 21.68 μm but no more than 48.2 μm apart. Before placing the instances, the software checks for cells named `WTAP` and `PSO1`, and if it finds any within the minimum gap (that is, 21.68 μm from the nearest well-tap cell in that row) it does not place any instances in that area.

  ```
  addWellTap -cell WTAP -incremental "WTAP PSO1" -maxGap 48.2
  ```

Related Topics

- Low Power Design chapter in the *Encounter User Guide*

- Substrate Biasing

- Placing the Design chapter in the *Encounter User Guide*
  - Adding Well-Tap Cells

checkFiller

```
checkFiller
     [-help]
     [-powerDomain power_domain_name]
     [-highlight {true| false}| -clearHighlight {true | false} ]
     [-adjacentFiller1 {true | false} | -reportGap microns]
     [-file fileName]
     [-area llx lly urx ury]
```

Checks for locations that are missing filler cells, after adding the cells with the `addFiller` command. Optionally, generates a report with details on sites that are missing filler cells.

If you run this command without any parameters, it reports the total number of sites in the core area that are missing filler cells.

Parameters

`-adjacentFiller1 {true | false}`

Checks for locations that are missing one-site-width fillers. To enable this parameter, you must first specify a one-site-width filler by using the following command:

```
setFillerMode -core list fillerCellList
```

`-area llx lly urx ury`

Specifies a bounding box for the area in which to check for missing filler cells.

Default: Entire core area

`-clearHighlight {true | false}`

Removes highlights added by the `-highlight` parameter.

`-file fileName`

Writes the output of the `checkFiller` command to the specified file.

`-help`

Outputs a brief description that includes type and default information for each `checkFiller` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man checkFiller.`

`-highlight {true | false}`

Highlights the locations where `checkFiller` reports missing filler cells.

`-powerDomain` *`power_domain_name`*

Checks for missing filler cells in the specified power domain.

the software checks for missing filler cells in all power domains if you do not specify this parameter.

`-reportGap` *`microns`*

Checks only for gaps of the specified size—ignores any other gap, whether larger or smaller.

Examples

The following command checks the entire core area after running `addFiller` and reports the number of sites that are missing filler cells:

```
checkFiller
```

The software reports the following:

```
Total number of unoccupied sites found: 85
```

checkPlace

```
checkPlace
     [-checkPinAccess | -noCheckPinAccess]
     [-clearMarker]
     [-honorPrerouteForFiller]
     [-ignoreOutOfCore]
     [-ioPinBlockage]
     [-noHardFence]
     [-noPreplaced]
     [-noHalo]
     [violationReportFileName]
```

Checks FIXED and PLACED cells for violations, adds violation markers to the design display area, and generates a violation report. Use the Violation Browser to see the violation markers.

Rules as to what constitutes a placement violation are different for FIXED and PLACED cells. For example, FIXED cells in obstruction areas are not considered to be violations, but PLACED cells are considered to be violations.

This command considers preplaces instances and macros as FIXED.

Note: The utilization value reported by checkPlace and the density value reported by timeDesign might differ because checkPlace considers padding when calculating utilization but timeDesign does not consider padding when calculating density.

You can delete padding by running the deleteAllCellPad command before running checkPlace.

The checkPlace command checks for the following violations:

- Region and fence violations (Instances placed outside their region or fence)
- Out of core area violations (Instances placed partly or wholly outside the core area)
- Not placed on placement grid (Instances not on the placement grid, or macros not on the manufacturing grid)
- Row orientation violations (Instances whose orientation is illegal for their row; may short power to ground)
- Tech site violations (Instances placed on tech sites not matching their cells and instances that do not have rows for their tech sites)
- Overlapping with other instance (Placed instances that overlap with other instances)
- Pin access violations (Instances with at least one signal pin that does not have routing access due to pre-wires)

- Orientation violations (Instances whose orientation is not legal based on its cell symmetry rules)
- Routing/placement blockage violations in the core area (Instances placed in an obstruction created by pre-wires or routing blockage)

  The command does not check fixed macros.
- Placement blockage violations in the core area (for example, placed instances in an obstruction created by prerouted wires or a routing blockage, placed instances that overlap a placement blockage area).

  Note: The command does not check fixed instances overlapping placement or routing blockages by default.
- Site orientation violations (Instances whose orientation does not match the tech site orientation)
- Not of fence violations (Instances placed inside a fence to which it does not belong)

In addition to checking for violations, the command reports the following:

- Number of placed instances in the design
- Placement density, in terms of used sites over available sites

Parameters

`-checkPinAccess | -noCheckPinAccess`

Specifies whether to check for pin access violations.
Default: `-checkPinAccess`

`-clearMarker`

Removes the violation markers from a previous `checkPlace` run, without rerunning `checkPlace`.

`-honorPrerouteForFiller`

Checks for placement blockage violations on physical instances placed under preroutes or specified by `setPrerouteAsObs`.

By default, `checkPlace` does not treat filler cells placed under preroutes as violations, even though they are placement/routing blockages; however, it does treat standard cells placed under preroutes as violations.

`-ignoreOutOfCore`

Skips instances outside the core design area.

`-ioPinBlockage`

Instructs legalization routines to treat top-level I/O pins as fixed metal obstructions while placing standard cell instances at the block level. There is a minor run-time penalty with this parameter, so it is not turned on by default.

`-noHalo` Skips block halos. Uses the place-and-route boundary for the placement overlap check.

`-noHardFence`

Ignores fence and region placement constraints.

`-noPreplaced` Does not check fixed instances. When this parameter is specified, `checkPlace` treats fixed instances as placement blockages.

`violationReportFileName`

Specifies the report file. If you do not specify this parameter, the command displays a summary report.

The report file is more detailed than the summary report, and lists every instance with a violation.

Related Topics

- Tools Menu chapter in the *Encounter Menu Reference*
  - "Violation Browser"

clearDeCapCellCandidates

```
clearDeCapCellCandidates
```

Clears all available cells for decoupling capacitance insertion.

Parameters

None

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - "Adding Decoupling Capacitance"

clearScanDisplay

`clearScanDisplay [`*`scanChainName`*`]`

Removes scan chains from the display.

Parameters

`scanChainName`	Specifies the scan chain to remove from the display. *Default:* Removes all scan chains from the display.

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

clearSpareCellDisplay

`clearSpareCellDisplay`

Removes spare cell instances from the display.

Parameters

None

clonePlace

`clonePlace`

Places all clone instances with relative location and relative orientation according to the master partition instances. Adjusts instance orientation based on the clone orientation.

Parameters

None

Command Order

Use this command after running `placeDesign`.

clusteringPlace

```
clusteringPlace
     [-cluRatio numberOfInsts]
     [-outfile fileName]
```

Bundles the netlistClustering and placeDesign commands as one command, and accepts parameters of both. This ensures that netlist clustering is followed immediately by placement.

You can use this command after importing the design. The `restoreRoute` command can also restore saved trial routing data for a clustered netlist.

Important

The only operations that can be done with a clustered netlist are trial routing and placement. You must uncluster the netlist before running any other operations. Use netlistUnclustering to uncluster the netlist.

Parameters

`-cluRatio` *numberOfInstances*

Specifies the average number of instances in a cluster. For example, if you specify `-cluRatio 4`, the software reduces a netlist with 2 million instances to clustered netlist with 500,000 instances.

`-outfile` *fileName*

Specifies the filename for the cluster information.

Default: If you do not specify this parameter, it is *cellName*`.clu.gz`.

congOpt

```
congOpt
     [-nrIterInCongOpt numberOfIterations | -maxCPUTimeInCongOpt hours]
```

Reduces congestion after placement in an iterative way. The design must be legally placed. If you use `congOpt` on a design that is not legally placed, the software does not run the refinement step during congestion optimization, and instead exits.

The software stops when it cannot make any more improvements.

Use this command after placing the design, or after global placement, timing optimization, or clock tree synthesis (CTS), when routability becomes the ultimate goal.

Parameters

`-maxCPUTimeInCongOpt` *`hours`*

Specifies the maximum CPU time in congestion optimization, in hours. This parameter overrides the `-nrIterInCongOpt` parameter: The software runs multiple iterations if necessary to optimize the design, even if it the number of iterations is greater than the number specified.

`-nrIterInCongOpt` *`numberOfIterations`*

Specifies the total number of iterations in congestion optimization.
Default: `1`

createSpareModule

```
createSpareModule
     -moduleName moduleName
     -cell {cellName [number] [cellName [number] ] …}
     [-clock netName]
     [-reset netName:pinName [pinName] …]
     [-tie {tieCellName [tieCellname] } ]
     [-tieLo {pinList | *} ]
     [-useCellAsPrefix]
```

Creates a spare module made up of the specified cells.

Parameters

-cell {*cellName* [*number*] [*cellName* [*number*]] …}

Specifies the names of the cells, and the number of each, to add to the spare module. If you do not specify the number of cells to add, the `setTieHiLoMode -maxFanOut` parameter controls the number of cells.

-clock *netName*

Specifies a clock net to connect to the clock pins of the instances in the spare module.

-moduleName *moduleName*

Specifies a name for the spare module.

-reset *netName*:*pinName* [*pinName*]

Specifies a net and the pins to connect to the reset pins of sequential spare cells in the spare module. If you do not specify this parameter, these reset pins are not connected.

-tie *tieCellName* [*tieCellName*]

Specifies tie-high and tie-low cells to add to the spare module. The software recognizes whether the pins are high or low.

-tieLo {*pinList* | *}

Specifies the pins to connect to the specified tie-low cell. If no tie-low cells are specified, connects the pins to `1'b0`.

Use the following format for the pin list:

```
<pinList>   :: {cellName: <pin>} …
<pin>       :: pinName |*
```

`-useCellAsPrefix`	Adds the master cell name as a prefix to the name of gates that are instantiated inside the module. The software adds `spr_gate_`*n* as a prefix by default; when you specify `-useCellAsPrefix`, it prepends the master cell name to the default prefix.

Examples

The following command creates spare module `mod1` with cells `cellA`, `cellB`, and `tiehi4`:

```
createSpareModule -cell {cellA cellB} -tie tiehi4 -moduleName mod1
```

The following command creates spare module `mod1` with the following gates: `and_spr_gate_1, and_spr_gate_2, nand_spr_gate_1, nor_spr_gate_1`:

```
createSpareModule -cell {and 2 nand nor} -moduleName mod1 -useCellAsPrefix
```

Related Topic

- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*

deleteAllCellPad

`deleteAllCellPad`

Removes placement padding specified by `specifyCellPad` for all cells. If you run this command before running `checkPlace`, `checkPlace` does not consider the cell padding when calculating effective utilization. For more information, see checkPlace on page 704 and specifyCellPad on page 807.

Parameters

None

Related Topic

- "Placing the Design" chapter in the *Encounter User Guide*
  - Adding Padding

deleteAllScanCells

```
deleteAllScanCells
```

Unassigns all assigned scan cells.

Parameters

None

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

deleteDeCap

`deleteDeCap`

Deletes all decoupling capacitance cells.

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - "Adding Decoupling Capacitance"

Parameters

None

deleteFiller

```
deleteFiller
     {-inst instanceName | [-cell fillerCellName ...] | [-prefix prefix]}
     [-area x1 y1 x2 y2]
     [-deleteFixed {true | false}]
```

Removes physical cell instances, such as filler cells, end-cap cells, and well-tap cells, from standard cell rows.

Important

Some `deleteFiller` parameters can also be specified with `setFillerMode`. However, the parameters you specify with `deleteFiller` apply to the current `deleteFiller` run only, and supersede parameters set by `setFillerMode`. For more information, see `setFillerMode`.

Parameters

`-area` *x1 y1 x2 y2*

Specifies the coordinates of the area in which to delete filler cells. Instances that cross the area boundary are not deleted.
Default: The core box that encloses all core rows

Note: You can specify this parameter with this command or with `addFiller` only, and not with `setFillerMode`.

x1 Specifies the area's lower left point on the x axis.

y1 Specifies the area's lower left point on the y axis.

x2 Specifies the area's upper right point on the x axis.

y2 Specifies the area's upper right point on the y axis.

`-cell` *cellName*

Specifies a cell name, or list of cells, to delete. Separate cell names with spaces.

Note: Cells specified by this parameter override the list of cells provided by the `-core` parameter of `setFillerMode`.

`-deleteFixed {true | false}`

Deletes filler cells marked `FIXED`.
Default: `true`. End-cap and well-tap cells are marked `FIXED`, so `deleteFiller` does not delete them by default.

`-prefix` *`prefix`*

Specifies the prefix of the cell instances to delete.

For example, to delete all filler cell instances with the prefix `FILL`, type the following command:

```
deleteFiller -prefix FILL
```

The software will remove all physical instances whose name starts with `FILL`, including instances named `FILL_1`, `FILLTEMP_1`, and so on.

`-inst` *`instanceName`*

Specifies the instance of the cell to delete. You can use wildcards (`*?`) with this parameter.

Note: You can specify this parameter with this command or with `addFiller`, and not with `setFillerMode`.

Examples

- The following command deletes all physical-only cells from the design:

  ```
  deleteFiller -cell xyz
  ```

- The following command deletes only the `xyz` cells whose instance names have an `wtap_even` prefix. If well-tap cells in even rows have a wtap_even prefix and well-tap cells in odd rows have a different prefix, the cells in rows 2, 4, 6, and so on are deleted.

  ```
  deleteFiller -cell xyz -prefix wtap_even
  ```

- The following command deletes all added cell instances with the prefix instance name `FILLER`:

  ```
  deleteFiller -prefix FILLER
  ```

- The following command deletes all instances of filler cell `FILL2`, end-cap `ecp17`, and well-tap `wtp5`:

  ```
  deleteFiller -cell FILL2 ecp17 wtp5
  ```

- The following command deletes all instances of given the spacer cells from the design:

  ```
  deleteFiller -cell FILL64 FILL32 Fill8 FILL16 FILL2 FILL1
  ```

- The following command deletes all well-tap and end-cap cell instances from the design:

```
deleteFiller -cell WELLTAP PRECAP POSTCAP
```

deleteInstPad

```
deleteInstPad [inst_name | -all]
```

Deletes the padding for the specified instances, or all instance-based padding, in the design.

Parameters

inst_name	Deletes the padding on the specified instance.
-all	Deletes all instance-based padding.

Related Topic

- "Placing the Design" chapter in the *Encounter User Guide*
  - Adding Padding

deleteScanCell

`deleteScanCell` *`cellName`*

Removes the scan cell specification from a library cell.

Parameters

`cellName`	Specifies the cell from which to remove the scan specification.

Example

The following command removes the scan specification from the cell named FF_SCAN:

```
deleteScanCell FF_SCAN
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

deleteScanChain

`deleteScanChain` *`scanChainName`*

Removes a scan chain specification from the design.

Tip

When you are debugging a new scan chain specification, it is useful to delete all scan chains and start over.

Parameters

`scanChainName`	Specifies the name of the scan chain to remove from the design. If you do not specify a scan chain name, the command deletes all scan chains.

Example

The following commands load the scan information from a DEF file, remove the scan chain ABC, and output the scan information to a DEF file:

```
defIn -scanChain
deleteScanChain ABC
defOut -scanChain
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

deleteScanChainPartition

```
deleteScanChainPartition
     {-all  |  -partition  partitionName1 partitionName2 ...}
```

Deletes scan chain partitions. The partitions are used to group scan chains so elements in compatible scan chains can be swapped. The scan chains are then reordered on a per-chain basis.

Specify scan chain partitions with the following command: specifyScanChainPartition.

Parameters

`-all`	Deletes all partitions from compatible groups.
`-partition` *`partitionName1 partitionName2 …`*	Specifies the partitions to delete.

Example

The following command deletes partition `clk1Domain`:

```
deleteScanChainPartition clk1Domain
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

deleteSpareModule

`deleteSpareModule` *`moduleName`*

Deletes an instantiation of a spare module from the netlist.

Parameters

`moduleName`	Specifies the instantiation to delete.

Example

The following command deletes spare module instantiation `spr_6`:

```
deleteSpareModule spr_6
```

Related Topic

- "Placing the Design" chapter in the *Encounter User Guide*
  - Placing Spare Cells and Spare Modules

deleteTieHiLo

```
deleteTieHiLo
     [-cell tieCellName | "listOfTieCellNames"]
     [-prefix prefixName]
```

Deletes placed tie cells from the netlist and reconnects the tie input pins back to `1'b1` and `1'b0`.

Parameters

`-cell tieCellName | "listOfTieCellNames"`
: Specifies the cell names to delete. You can specify a single cell name, or a list of cell names, which must be surrounded by quotation marks.

`-prefix prefixName`
: Specifies the prefix of the instances to delete.

Note: If you do not specify any parameters, the command uses the global cell names if they were set previously with the `setTieHiLoMode` command.

Example

The following command deletes all placed tie-off cell instances with the prefix `ABC`:

```
deleteTieHiLo -prefix ABC
```

displayScanChain

```
displayScanChain [scanChainName]
```

Displays a scan chain. You must load the scan information, place the design, and view the design in the Placement view.

Parameters

scanChainName	Specifies the scan chain. *Default:* If you do not specify this parameter, the software displays all scan chains.

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

displaySpareCell

```
displaySpareCell
```

Displays the results of placing the spare instances. You must specify the spare cell types or spare instances before placing the design.

Parameters

None

getDensityMapMode

```
getDensityMapMode
     [-gridInMicron]
     [-gridInRow]
     [-quiet]
     [-threshold]
```

Returns the following information about `setDensityMapMode` parameters in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Source of current value (default, user)

If you do not specify a parameter, the software displays values for all of the `setDensityMapMode` mode parameters.

See `setDensityMapMode` for parameter descriptions.

Parameters

`parameter_names`	Returns information for the specified parameter. You can specify one or more parameters.
`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setDensityMapMode` parameters in Tcl list format.

Examples

- The following command returns the current setting for the `-gridInMicron` parameter in Tcl list format:

  ```
  getDensityMapMode -gridInMicron -quiet
  ```

  The software returns the following information:

  ```
  50
  ```

- The following command returns the current settings for the `-gridInMicron` and `-gridInRow` parameters:

  ```
  getDensityMapMode -gridInMicron -gridInRow
  ```

  The software returns the following information:

  ```
  -gridInMicron 50          # float, default=50, min=0.000000, max=1000.000000
  -gridInRow 10             # int, default=10, min=1, max=100
  {gridInMicron 50} {gridInRow 10}
  ```

- The following command returns the current settings for the `-core` and `-corePrefix` parameters in Tcl list format:

  ```
  getDensityMapMode -gridInMicron -gridInRow -quiet
  ```

  The software returns the following information:

  ```
  {gridInMicron 50} {gridInRow 10}
  ```

- The following command returns the current settings for all `setDensityMapMode` parameters in Tcl list format:

  ```
  getDensityMapMode -quiet
  ```

  The software returns the following information:

  ```
  {gridInMicron 50} {gridInRow 10} {threshold 0.75}
  ```

getFillerMode

```
getFillerMode
     [-core]
     [-corePrefix]
     [-createRows]
     [-deleteFixed]
     [-doDRC]
     [-ecoMode]
     [-fitGap]
     [-honorPrerouteAsObs]
     [-merge]
     [-minHoleCheck]
     [-quiet]
```

Returns the following information about `setFillerMode` parameters in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays values for all of the `setFillerMode` mode parameters.

Parameters

`parameter_names`	Returns information for the specified parameters. You can specify one or more parameters. See `setFillerMode` for parameter descriptions.
`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setFillerMode` parameters in Tcl list format.

Examples

- The following command returns the current setting for the `-core` parameter in Tcl list format:

  ```
  getFillerMode -core -quiet
  ```

  The software returns the following information:

  ```
  {}
  ```

- The following command returns the current settings for the `-core` and `-corePrefix` parameters:

  ```
  getFillerMode -core -corePrefix
  ```

  The software returns the following information:

  ```
  -core {}                      # string, default=""
  -corePrefix FILLER            # string, default=FILLER
  {core {}} {corePrefix FILLER}
  ```

- The following command returns the current settings for the `-core` and `-corePrefix` parameters in Tcl list format:

  ```
  getFillerMode -core -corePrefix -quiet
  ```

  The software returns the following information:

  ```
  {core {}} {corePrefix FILLER}
  ```

- The following command returns the current settings for all `setFillerMode` parameters in Tcl list format:

  ```
  getFillerMode -quiet
  ```

  The software returns the following information:

  ```
  {core {FILL16BWP FILL4BWP FILL1BWP}} {corePrefix FILL} {createRows true}
  {deleteFixed true} {doDRC true} {ecoMode false} {fitGap false}
  {honorPrerouteAsObs false} {merge false} {minHoleCheck false}
  ```

getPlaceMode

```
getPlaceMode
     [-blockedShifterCols]
     [-blockedShifterRows]
     [-checkPinLayerForAccess]
     [-checkRoute]
     [-clkGateAware]
     [-colShiftersOnly]
     [-congEffort]
     [-dividedShifterCols]
     [-dividedShifterRows]
     [-doCongOpt]
     [-doRPlace]
     [-fixedShifter]
     [-fp]
     [-hardFence]
     [-honorImplantSpacing]
     [-honorSoftBlockage]
     [-ignoreScan]
     [-ignoreSpare]
     [-maxDensity]
     [-maxRouteLayer]
     [-moduleAwareSpare]
     [-modulePadding]
     [-modulePlan]
     [-padFixedInsts]
     [-padForPinNearBorder]
     [-placeIoPins]
     [-powerDriven]
     [-preserveRouting]
     [-reorderScan]
     [-rmAffectedRouting]
     [-swapEEQ]
     [-tdInstPadding]
     [-timingDriven]
     [-viaInPin]
     [-wireLenOptEffort]
     [-quiet]
```

Displays the following information about `setPlaceMode` parameters in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the Encounter software displays values for all of the `setPlaceMode` mode parameters.

Parameters

`parameter_names`	Returns information for the specified parameters. You can specify one or more parameters. See `setPlaceMode` for parameter descriptions.
`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setPlaceMode` parameters in Tcl list format.

Examples

- The following command returns the current setting for the `-blockedShifterCols` parameter in Tcl list format:

  ```
  getPlaceMode -blockedShifterCols -quiet
  ```

  The software returns the following information:

  ```
  false
  ```

- The following command returns the current settings for the `-blockedShifterCols` and `-blockedShifterRows` parameters:

  ```
  getPlaceMode -blockedShifterCols -blockedShifterRows
  ```

  The software returns the following information:

  ```
  -blockedShifterCols false               # bool, default=false
  -blockedShifterRows false               # bool, default=false
  {blockedShifterCols false} {blockedShifterRows false}
  ```

- The following command returns the current settings for the `-blockedShifterCols` and `-blockedShifterRows` parameters in Tcl list format:

  ```
  getPlaceMode -blockedShifterCols -blockedShifterRows -quiet
  ```

  The software returns the following information:

  ```
  {blockedShifterCols false} {blockedShifterRows false}
  ```

- The following command returns the current settings for all `setPlaceMode` parameters in Tcl list format:

```
getPlaceMode -quiet
```

The software returns the following information:

```
{blockedShifterCols false} {blockedShifterRows false} {checkPinLayerForAccess
1} {checkRoute false} {clkGateAware false} {colShiftersOnly false} {congEffort
medium} {dividedShifterCols false} {dividedShifterRows true} {doRPlace true}
{fixedShifter false} {fp false} {hardFence true} {honorImplantSpacing false}
{honorSoftBlockage false} {ignoreScan true} {ignoreSpare true} {maxDensity -1}
{maxRouteLayer {}} {maxShifterColDepth 9999} {maxShifterDepth 9999}
{maxShifterRowDepth 9999} {moduleAwareSpare false} {modulePadding {}}
{modulePlan true} {padFixedInsts false} {padForPinNearBorder false}
{placeIoPins true} {powerDriven false} {preserveRouting false} {reorderScan
true} {rmAffectedRouting false} {rowShiftersOnly false} {strictShifterSide
false} {strictShifterSpot false} {swapEEQ false} {tdInstPadding false}
{timingDriven true} {viaInPin false} {wireLenOptEffort none}
```

getScanReorderMode

```
getScanReorderMode
     [-allowSwapping]
     [-clkAware]
     [-keepPDPorts]
     [-keepPort]
     [-preferH]
     [-preferV]
     [-scanEffort]
     [-skipMode]
     [-quiet]
```

Returns the following information about `setScanReorderMode` parameters in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Source of current value (default, user)

If you do not specify a parameter, the Encounter software returns values for all of the `setScanReorderMode` mode parameters.

Parameters

`parameter_names`	Returns information for the specified parameter. You can specify one or more parameters. See setScanReorderMode on page 800 for parameter descriptions.
`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setScanReorderMode` parameters in Tcl list format.

Examples

- The following command returns the current setting for the `-keepPDPorts` parameter in Tcl list format:

```
getScanReorderMode -keepPDPorts -quiet
```

The software returns the following information:

```
false
```

- The following command returns the current settings for the `-keepPDPorts` and `-preferH` parameters:

  ```
  getScanReorderMode -keepPDPorts -preferH
  ```

  The software returns the following information:

  ```
  -keepPDPorts false   # bool, default=false
  -preferH false         # bool, default=false
  {keepPDPorts false} {preferH false}
  ```

- The following command returns the current settings for the `-keepPDPorts` and `-preferH` parameters in Tcl list format:

  ```
  getScanReorderMode -keepPDPorts -preferH -quiet
  ```

  The software returns the following information:

  ```
  {keepPDPorts false} {preferH false}
  ```

- The following command returns the current settings for all `setScanReorderMode` parameters in Tcl list format:

  ```
  getScanReorderMode -quiet
  ```

  The software returns the following information:

  ```
  {allowSwapping false} {clkAware false} {keepPDPorts true} {keepPort (false)}
  {preferH false} {preferV false} {scanEffort medium} {skipMode skipNone}
  ```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

getTieHiLoMode

```
getTieHiLoMode
     [-cell]
     [-createHierPort]
     [-honorDontTouch]
     [-maxDistance]
     [-maxFanout]
     [-prefix]
     [-reportHierPort]
     [-quiet]
```

Returns the following information about `setTieHiLoMode` parameters in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the Encounter software displays values for all of the `setTieHiLoMode` mode parameters.

Parameters

`parameter_names`	Returns information for the specified parameters. You can specify one or more parameters. See `setTieHiLoMode` for parameter descriptions.
`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setTieHiLoMode` parameters in Tcl list format.

Examples

- The following command returns the current setting for the `-cell` parameter in Tcl list format:

  ```
  getTieHiLoMode -cell -quiet
  ```

The software returns the following information:

```
{}
```

- The following command returns the current settings for the `-cell` and `-honorDontTouch` parameters:

  ```
  getTieHiLoMode -cell -honorDontTouch
  ```

  The software returns the following information:

  ```
  -cell {}                          # string, default=""
  -honorDontTouch false             # bool, default=false
  {cell {}} {honorDontTouch false}
  ```

- The following command returns the current settings for the `-cell` and `-honorDontTouch` parameters in Tcl list format:

  ```
  getTieHiLoMode -cell -honorDontTouch -quiet
  ```

  The software returns the following information:

  ```
  {cell {}} {honorDontTouch false}
  ```

- The following command returns the current settings for all `setTieHiLoMode` parameters in Tcl list format:

  ```
  getTieHiLoMode -quiet
  ```

  The software returns the following information:

  ```
  {cell {}} {honorDontTouch false} {maxDistance 0} {maxFanout 0}
  ```

netlistClustering

```
netlistClustering
     [-cluRatio numberOfInstances]
     [-outfile fileName]
```

Clusters a flattened netlist, generates a clustered netlist, and stores the cluster information to a file. A clustered netlist uses the same logical hierarchy as the original netlist.

To run netlist clustering and placement in one command, use the clusteringPlace command.

Important

The only operations that can be done with a clustered netlist are trial routing and placement. You must uncluster the netlist before running any other operations. Use netlistUnclustering to uncluster the netlist.

Parameters

`-cluRatio` *numberOfInstances*

Specifies the average number of instances in a cluster. For example, if you specify `-cluRatio 4`, the software reduces a netlist with 2 million instances to clustered netlist with 500,000 instances.

`-outfile` *fileName*

Specifies the filename for the output of this command.

Default: *cellName*`.clu.gz`.

Command Order

Use this command after running `designImport` and `loadFPlan`, and before running `placeDesign` or `restorePlace`.

netlistUnclustering

`netlistUnclustering [`*`fileName`*`]`

Restores a clustered netlist after placement, so that it is unclustered and all the original instances are placed where the clusters were placed.

Parameters

`fileName`	Specifies the filename for the cluster information. *Default:* *`cellName`*`.clu.gz`.

placeDesign

```
placeDesign
     [-ilm]
     [-inPlaceOpt]
     [-incremental]
     [-noPrePlaceOpt]
```

Places standard cells based on the global settings for placement, RC extraction, timing analysis, and trial routing. Also performs the following operations:

- Pre-placement optimization
- Scan tracing and scan reordering
- Placement optimization during the placement process
- Incremental placement on an already placed design

By default, `placeDesign` performs timing-driven placement with scan reordering, except during prototyping, when it runs in non-timing-driven mode. To change this behavior, run setPlaceMode with the following parameters prior to running `placeDesign`:

- `-reorderScan false`
- `-timingDriven false`

Or, in prototyping mode:

- `-fp`

If there are no timing constraints, `placeDesign` assumes non-timing-driven mode.

Note: This command supports multiple-CPU processing. For information, see Multiple-CPU Processing Commands and Accelerating the Design Process by Using Multiple-CPU Processing.

The `placeDesign` command honors settings for the following commands with the exceptions noted in the table below:

Command	Exceptions/Notes
`setAnalysisMode`	■ Does not honor `-checkType hold`. If you specify `-checkType hold`, issues a warning and assumes `-checkType setup` mode. ■ Does not honor `-clockPropagation sdcControl`. If you specify `-clockPropagation sdcControl`, assumes `-clockPropagation forcedIdeal`. ■ Does not honor `-clkSrcPath true`. If you specify `-clkSrcPath true`, assumes `-clkSrcPath false`.
`setClockDomains`	All parameters honored The `placeDesign` command removes the loaded clock tree specification file from memory after placement, because timing optimization might make instance changes that invalidate the file. You must reload the clock tree specification file if you want to run `ckSynthesis` or `ckCloneGate`.
`setExtractRCMode`	Does not honor `-detail`. If you specify `-detail`, issues a warning and assumes `-default` mode.
`setOptMode`	All parameters honored
`setPlaceMode`	If you specify `-clkGateAware`, you must specify a clock specification file before running `placeDesign`.
`setScanReorderMode`	You must specify scan chain information or import a scan DEF file or the software issues a warning message and skips scan reordering.
`setTrialRouteMode`	A subset of parameters is honored

Parameters

`-ilm`	Honors Interface Logic Models (ILMs) in timing-driven placement. Cells are placed at the top level at an optimum distance to the interface logic within the block. **Note:** You cannot use this parameter with `-incremental` or `-inPlaceOpt`: an error occurs. **Note:** The `-ilm` parameter is obsolete. This command detects ILMs by default.
`-incremental`	Runs placement incrementally. This parameter works only when the design is already placed. **Note:** Using this parameter defaults to `-noPrePlaceOpt`.
`-inPlaceOpt`	Performs timing-driven placement with optimization. Since this parameter performs optimization with `optDesign -preCTS`, you must specify the optimization parameter with `setOptMode` before you run `placeDesign`. The netlist is restructured as a result. While this parameter improves timing in most cases, it causes an increase in run time.  *Important* If you run the following command (or you have not loaded timing constraints) before running `placeDesign -inPlaceOpt`, the in-place optimization flow is disabled: `setPlaceMode -timingDriven false`
`-noPrePlaceOpt`	Disables pre-place optimization during the placement run. By default, the command removes all cells with the specified buffer footprint (same as with the `deleteBufferTree` command), except when using the `-incremental` parameter.

Command Order

Use the `placeDesign` command after performing general floorplanning, including macro placement, regioning (if applicable), and power planning on the design. The design could be in the unplaced state, or in a state that the existing placement is to be replaced.

- To run timing-driven placement (the default), make sure the timing constraints are loaded in the design.

- To run scan reorder (the default), specify the scan chain or import the scan DEF data before running this command.

Examples

- The following commands perform non-timing-driven placement:

```
setPlaceMode -timingDriven false
placeDesign
```

- The following commands disable scan chain reordering:

```
setPlaceMode -reorderScan false
placeDesign
```

- The following command disables the pre-placed buffer tree removal:

```
placeDesign -noPrePlaceOpt
```

- The following command enables the in-place optimization flow:

```
placeDesign -inPlaceOpt
```

- After the design is placed, the following command performs incremental placement, and disables the pre-placed buffer tree removal:

```
placeDesign -incremental
```

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*

placeInstance

```
placeInstance
     instanceName
     x1 y1
     [orientation]
     [-fixed | -placed]
```

Places a leaf instance in the core design area.

Note: If you place a master instance blackbox to non-R0 orientation, the new non-R0 orientation will be converted to R0 orientation. For more information, see Handling of Blackboxes with Non-R0 Orientation in the “Partitioning the Design” chapter of the *Encounter User Guide*.

Parameters

instanceName	Specifies the leaf instance name.
x1	Specifies the lower-left x coordinate of the instance on the x axis.
y1	Specifies the lower-left y coordinate of the instance on the y axis.
orientation	Specifies the orientation of the instance. Specify one of the following values: `R0`, `R90`, `R180`, `R270`, `MX`, `MX90`, `MY`, or `MY90`.
`-fixed | -placed`	Specifies that the cell being placed is in `fixed` or `placed` status. *Default*: `-fixed`

Example

The following command places the instance `SH22/I40` with mirrored axis orientation at location `1500.3`, `980.3` in the floorplan. The placement program does not move the instance:

```
placeInstance SH22/I40 1500.3 980.3 MY
```

placeJtag

```
placeJtag
     -nrRow nrRow
     [-nrRowTop nrRowTop]
     [-nrRowBottom nrRowBottom]
     [-nrRowLeft nrRowLeft]
     [-nrRowRight nrRowRight]
     [-ioNetWeight netWtValue]
     [-blockNetWeight netWtValue]
     [-hardMacro]
     [-contour]
     [-areaIo]
     [-ignoreScan]
     [-orientBottom orientation]
     [-orientLeft orientation]
     [-orientRight orientation]
     [-orientTop orientation]
```

Places JTAG cells. Use this command for JTAG and other cells where placement is required to be close to the perimeter I/Os.

Parameters

`-areaIo`
: Considers the area around I/O cells as a legal placement location. When specifying this parameter, the `-ioNetWeight` parameter is applied to the nets that are connected to the Area I/O cells.

`-blockNetWeight` *`netWtValue`*
: Specifies the weight for nets connecting to blocks. This allows `placeJtag` to place more critical block nets in a smaller area.

  Default: `20`
  Maximum value: `512`

`-contour`
: Creates an outline along the boundary of the chip, with its width based on the number of standard cell rows, as specified by the `-nrRow` parameter(s). You can specify different values for `-nrRowTop`, `-nrRowBottom`, `-nrRowLeft`, and `-nrRowRight`.

`-hardMacro`
: Includes the boundaries of the hard macros in the JTAG placement.

`-ignoreScan`

Ignores all scan-in pins, scan enable pins, clock pins, and asynchronous control pins.

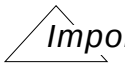
Important

`placeJtag` relies on the timing library to provide information, such as what pins are scan-in, scan enable, clk, and async control. If the timing library does not have the scan-in and scan enable information, `placeJtag` is not able to ignore them. In this case, before you run `placeJtag`, you must specify those terms using the `specifyScanCell` command.

`-ioNetWeight` *`netWtValue`*

Specifies the weight for nets connecting to I/Os. This allows `placeJtag` to place more critical I/O nets in a smaller area.

Default: If you do not specify this parameter, the weight is `100`. The maximum is `512`.



Important

`-ioNetWeight` applies to the nets that are connected to any I/O cell, whether it is a standard peripheral I/O cell or an Area I/O cell. This behavior is different than `-blockNetWeight`, which specifies the weight of the nets connected to blocks separately from the I/O nets.

`-nrRow` *`nrRow`*

Specifies the number of standard cell rows from the chip's perimeter where JTAG cells will be placed. This parameter creates a ring-shaped placement area for JTAG cells. The dimension of the top and bottom of the ring is the standard cell height. The dimension of the sides of the ring is the `nrRow` times the standard row height.

`-nrRowBottom` *`nrRowBottom`*

Specifies the number of standard cell rows at the bottom of the core area to place JTAG logic.

`-nrRowLeft` *`nrRowLeft`*

Specifies the number of standard cell row heights at the left of the core area to place JTAG logic.

`-nrRowRight` *`nrRowRight`*

Specifies the number of standard cell row heights at the right of the core area to place JTAG logic.

`-nrRowTop` *`nrRowTop`*

Specifies the number of standard cell rows at the top of the core area to place JTAG logic.

`-orientBottom` *`orientation`*

Considers the specified orientation at each side first and then looks for other legal orientations. Specify one of the following values: `R0 R90 R180 R270 MX MX90 MY MY90`.

`-orientLeft` *`orientation`*

Considers the specified orientation at each side first and then looks for other legal orientations. Specify one of the following values: `R0 R90 R180 R270 MX MX90 MY MY90`.

`-orientRight` *`orientation`*

Considers the specified orientation at each side first and then looks for other legal orientations. Specify one of the following values: `R0 R90 R180 R270 MX MX90 MY MY90`.

`-orientTop` *`orientation`*

Considers the specified orientation at each side first and then looks for other legal orientations. Specify one of the following values: `R0 R90 R180 R270 MX MX90 MY MY90`.

Examples

The following command places the JTAG instances within four standard cell rows at the top and bottom of the chip and within four times the standard cell row height at the sides of the chip:

```
placeJtag -nrRow 4
```

The following command places the JTAG instances within ten standard cell rows at the top and bottom of the chip and within ten times the standard cell row height at the sides of the chip. It increases the I/O net weight to 256.

```
placeJtag -nrRow 10 -ioNetWeight 256
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os
- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*

placePad

```
placePad
     padName
     x1 y1
     [orientation]
```

Places an I/O pad instance.

Parameters

x1	Specifies the lower left coordinate of the pad instance on the x axis.
y1	Specifies the lower left y coordinate of the pad instance on the y axis.
orientation	Specifies the orientation of the pad cell. Specify one of the following values: R0, R90, R180, R270, MX, MX90, MY, and MY90. *Default:* R0
padName	Specifies the pad instance name.

Example

The following command places the I/O pad instance with mirrored axis orientation, rotated 90 degrees counterclockwise:

```
placePad SH14/I412 498.3 5000.8 MX90
```

placeSpareModule

```
placeSpareModule
     -moduleName moduleName
     {-stepx stepDistance -stepy stepDistance | -numModules integer}
     [-area x1 y1 x2 y2]
     [-channel
          {-maxWidth maxWidth}
          {-minWidth minWidth}
          {-minLen minLength}]
     [-offsetx offsetDistance]
     [-offsety offsetDistance]
     [-powerDomain powerDomainName]
     [-prefix prefix]
     [-util utilizationFactor]
```

Instantiates a created spare module, and places spare modules across the design. You can run this command before or after placement. If you run it after placement, it calls ecoPlace.

Parameters

`-area x1 y2 x2 y2`

Specifies the area in which to add spare modules.

x1 Specifies the area's lower left x coordinate.

y1 Specifies the area's lower left y coordinate.

x2 Specifies the area's upper right x coordinate.

y2 Specifies the area's upper right y coordinate.

`-channel` Places spare modules in thin channels between placement blockages.

`-maxWidth maxWidth`

Specifies the maximum channel width, in microns. The *maxWidth* determines whether the placeable area between two macro blocks is considered a channel.

`-minLen minLength`

Specifies the minimum channel length, in microns. The *minLength* must be greater than the specified *maxWidth.*

`-minWidth` *`minWidth`*

Specifies the minimum channel width, in microns. The *`minWidth`* should be greater than the width of the largest-space cell provided.

`-module` *`moduleName`*

Specifies the name of the spare module. This module must have been created using the `createSpareModule` command.

`-numModules` *`integer`*

Specifies the total number of spare modules to add. The software calculates how far apart to place the modules.

`-offsetx` *`offsetDistance`*

Specifies the offset along the x axis, in microns, from the left core boundary.

`-offsety` *`offsetDistance`*

Specifies the offset along the y axis, in microns, from the bottom core boundary.

`-powerDomain` *`powerDomainName`*

Specifies the power domain in which to place the spare modules.

`-prefix` *`prefix`*

Specifies a prefix to add to the automatically generated prefix for spare modules.

The software already uses the prefix `spr_`*`n`* for spare modules. If you specify `-prefix spareMod`, the instances of this module are named `spareMod_spr_1`, `spareMod_spr_2`, `spareMod_spr_3`, and so on.

`-stepx` *`stepDistance`*

Specifies the distance, in microns, between the instantiated spare modules along the x axis. The *`stepDistance`* must be greater than the spare module bounding box.

`-stepy` *`stepDistance`*

Specifies the distance, in microns, between the instantiated spare modules along the y axis. The *`stepDistance`* must be greater than the spare module bounding box.

`-util` *`utilizationFactor`*

Specifies the utilization factor of the spare modules.
Value Range: Greater than `0.0`; less than `1.0`

This parameter is obsolete. The obsolete parameter still works in this release, but to avoid this warning and to ensure compatibility with future releases, remove it from your script.

Command Order

Run this command after running `createSpareModule`.

Examples

The following command instantiates module `mod1` and places the instantiations at intervals of 50 microns along the x and y axis. It places the instantiations at an offset of 5 microns from the left core boundary and the bottom core boundary:

```
placeSpareModule -moduleName  mod1 -stepx 50 -stepy 50 -offsetx 5 -offsety 5
  -util 0.6
```

The following commands create spare modules between channels that are between 4.7 um and 10.0 um wide and at least 11.0 um long. The step size is 10 um in the x direction and 10 um in the y direction and the maximum utilization rate is 60 percent:

```
createSpareModule -moduleName SPGM -cell {DFFNSRX2 INVX4 \
  NOR2X2} -clock sysclk
placeSpareModule -moduleName SPGM -channel -minWidth 4.7 -maxWidth 10 \
  -minLength 11 -stepx 10 -stepy 10 -util 0.6
```

Related Topic

- “Placing the Design” chapter in the *Encounter User Guide*
  - Placing Spare Cells and Spare Modules

queryDensityInBox

```
queryDensityInBox
     x1 y1 x2 y2
```

Outputs placement density for a specific area in the design. Considers placement blockages and honors the metal layers specified by the `setPreRouteAsObs` command. This command is optimized for small area interactive queries. Use the queryPlaceDensity command to get the entire design density.

Parameters

x1	Specifies the area's lower left point on the x axis.
y1	Specifies the area's lower left point on the y axis.
x2	Specifies the area's upper right point on the x axis.
y2	Specifies the area's upper right point on the y axis.

queryPlaceDensity

`queryPlaceDensity`

Outputs standard cell density for the design, as seen by the global placer. If region or fence placement constraints exist on intermediate modules or inst-groups, then the command also reports the placement density separately for each.

For information on region and fence constraints, see Module Constraint Types in the "Floorplanning the Design" chapter of the *Encounter User Guide*.

Use the queryDensityInBox command to get the density for a specific area in the design.

Parameters

None

Example

The following in an example of an output message when running `queryPlaceDensity`, where `MODULE1`, `MODULE2`, and `MODULE3` are intermediate cells in the top cell:

```
*info: density for module 'MODULE1' = 0.219
    = stdcell_area 66.00 / alloc_area 301.00.
*info: density for module 'MODULE2' = 0.281
    = stdcell_area 66.00 / alloc_area 235.00.
*info: density for module 'MODULE3' = 0.518
    = stdcell_area 163250.00 / alloc_area 315370.69.
*info: density for the rest of the design = 0.000
    = stdcell_area 5168.00 / alloc_area 11505378.94.
```

refinePlace

```
refinePlace
     [-blockedShifterCols {true | false}]
     [-blockedShifterRows {true | false}]
     [-checkPinLayerForAccess {list_of_layer_numbers}]
     [-checkRoute {true | false}]
     [-dividedShifterCols {true | false}]
     [-dividedShifterRows {true | false}]
     [-fixedShifter {true | false}]
     [-hardFence {true | false}]
     [-honorImplantSpacing {true | false}]
     [-honorSoftBlockage {true | false}]
     [-ioPinBlockage {true | false}]
     [-padFixedInsts {true | false}]
     [-padForPinNearBorder {true | false}]
     [-preserveRouting {true | false}]
     [-rmAffectedRouting {true | false}]
     [-swapEEQ {true | false}]
     [-viaInPin {true | false}]
     [-wireLenOptEffort {none | medium | high}]
```

Corrects flawed cell locations and reports corrections or instance overlap problems. You do not need a fully placed design to run `refinePlace`.

To place unplaced instances, use the `ecoPlace` command.

Note: You can also use `setPlaceMode` to specify parameters for this command. If you use `setPlaceMode`, the parameters apply every time you run `refinePlace` in the current session, unless they are overwritten by the `refinePlace` settings.

Parameters

`-blockedShifterCols {true | false}`

Prevents standard cells from being placed near shifters on vertical edges of the power domain boundary.
Default: `false`

`-blockedShifterRows {true | false}`

Prevents standard cells from being placed near shifters on horizontal edges of the power domain boundary.
Default: `false`

`-checkPinLayerForAccess` *`list_of_layer_numbers`*

Specifies the layers to check for pin access.

If the design has pins on *metal1* and *metal2*, to check pin access for pins on both layers, type the following command:

```
refinePlace -checkPinLayerForAccess {1 2}
```

This command checks pin access for *metal1* pins from *metal2* and for *metal2* pins from *metal3*.

Default: `1`

`-checkRoute {true | false}`

Considers pre-routed `FIXED` signal wires and avoids creating DRC violations between `FIXED` signal wires and instance pins. When this parameter is not specified, `refinePlace` ignores `FIXED` signal wires.
Default: `false`

Note: Using this parameter might result in longer run time.

 *Tip*

If you specify `-ioPinBlockage`, you do not need to specify `-checkRoute` for signal pins because `refinePlace` will treat the signal pins similar to supply pins for blockages. However, signal wires still need this parameter.

`-dividedShifterCols {true | false}`

Forces the level shifters assigned to vertical edges of a power domain boundary to seek out the power domain corners. Otherwise, the shifters can be placed anywhere on those edges, usually trying to minimize wire length.
Default: `false`

`-dividedShifterRows {true | false}`

Forces the level shifters assigned to horizontal edges of a power domain boundary to seek out the power domain corners. Otherwise, the shifters can be placed anywhere on those edges, usually trying to minimize wire length.
Default: `true`

`-fixedShifter {true | false}`

Specifies whether to set the shifters to `FIXED` status after placing them.
Default: `false`

`-hardFence {true | false}]`

Specifies whether `refinePlace` honors or ignores the fence and region placement constraints while legalizing placed instances.
Default: `true` (`refinePlace` honors the constraints)

`-honorImplantSpacing {true | false}`

Makes the `refinePlace` command aware of implant layers. Specifying this parameter allows you to enforce minimum spacing rules between low voltage threshold (LVT) and high voltage threshold (HVT) cells. The spacing rules are enforced for cells in the same row.

To enable this parameter, complete the following steps:

- Specify the following in the LEF file:
  - Include implant layer geometries in the macro obstruction statements for standard cells.
  - Include a value for the minimum spacing between implant layers in the Layer (Implant) statement.

  For more information on LEF, see "LEF Syntax" in the *LEF/DEF Language Reference*.
- Insert filler cells between the LVT and HVT cells to provide proper spacing.

Default: `false`

`-honorSoftBlockage {true | false}`

Specifies whether soft placement blockages are considered as blockages during `refinePlace`. By default, soft placement blockages are ignored during `refinePlace`.
Default: `false`

Note: Soft blockages are different from density screens (partial blockages). Partial blockages are never considered during `refinePlace`.

`-ioPinBlockage {true | false}`

Instructs legalization routines to treat top-level I/O pins as fixed metal obstructions while placing standard cell instances at the block level. There is a minor run-time penalty with this parameter, so it is not turned on by default.
Default: `false`

Tip

If you specify `-ioPinBlockage`, you do not need to specify `-checkRoute` for signal pins because `refinePlace` will treat the signal pins similar to supply pins for blockages.

`-padFixedInsts {true | false}`

Specifies whether the `refinePlace` command honors cell padding on `FIXED` instances.
Default: `false`

`-padForPinNearBorder {true | false}`

Adds padding to a cell if any signal pin is near enough to the border to cause DRC violations with any metal geometry.
Default: `false`

`-preserveRouting {true | false}`

Preserves all routed wires.
Default: `false`. `refinePlace` deletes all routed wires.

`-rmAffectedRouting {true | false}`

Removes wires connected to moved cells.
Default: `false`. Removes all wires, whether connected cells were moved or not.

`-swapEEQ {true | false}`

Determines whether master cells can be replaced by EEQ cells during detailed placement to improve routability. After detailed placement, the software reports the number of replacements. EEQ cells are defined in the LEF file.
Default: `false`

`-viaInPin {true | false}`

Specifies that pin access checks assume that the router drops a via to a pin only if the via is fully contained inside the pin geometry.
Default: `false`

Specify this parameter only when the `-routeWithViaInPin` parameter is also specified for `setNanoRouteMode`. For information, see "setNanoRouteMode" on page 1280.

`-wireLenOptEffort {none | medium | high}`

Optimizes wire length by swapping cells. Optimization can reduce the total wire length up to two percent without degrading placement. In high-effort mode, it might add up to ten percent to the run time.
Default: `none` when `refinePlace` runs as a standalone command. When `refinePlace` runs as part of `placeDesign`, the default value is `medium`; when `refinePlace` runs as part of `optDesign` the default value is `none`.

Specify one of the following values:

`none`

Does not optimize wirelength.

`medium`

Reduces wirelength by approximately one percent.

`high`

Reduces wirelength by approximately two percent.

Command Order

Use this command after finding an error with the `checkPlace` command.

reportDeCap

`reportDeCap -area` *`xl yl x2 y2`*

Reports placed decoupling capacitance cells in the specified area.

Parameters

`-area` *`xl yl x2 y2`* Defines the area for which to report placed decoupling capacitance cells.

- *x1* Specifies the area's lower left x coordinate.
- *y1* Specifies the area's lower left y coordinate.
- *x2* Specifies the area's upper right x coordinate.
- *y2* Specifies the area's upper right y coordinate.

reportDeCapCellCandidates

`reportDeCapCellCandidates [-file outputFileName]`

Reports the available cells for decoupling capacitance insertion.

Parameters

`-file outputFileName`

Outputs a list of the available cells for decoupling capacitance insertion to the specified file. This file can then be used by the `addDeCapCellCandidates` command to define cell candidates in a different session, or after cell candidates have been deleted.

reportInstPad

```
reportInstPad {instance_name | -all}
```

Reports the padding for a specific instance, or all instance padding in the design.

Parameters

`-all`	Reports all instance padding in the design.
`instance_name`	Reports the padding for the specified instance.

Related Topic

- "Placing the Design" chapter in the *Encounter User Guide*
  - Adding Padding

reportJtagInst

```
reportJtagInst [-report fileName]
```

Displays JTAG information specified by the `specifyJtag` command or creates an output file that contains the information. For information on this command, see specifyJtag on page 809.

Parameters

`-report fileName`	Specifies the name of the JTAG report file. If you do not specify this parameter, the command prints the information to the screen without saving it in a file.

Example

The following command creates a JTAG report file:

```
reportJtagInst -report jtag.rpt
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os

reportDensityMap

```
reportDensityMap
     [-help]
     [-gridInMicron microns | -gridInRow numberRows]
     [-threshold density]
```

Generates a map that uses colors to represent placement density and generates a placement density report. If you run this command without parameters, it generates a map with a threshold of 0.75 and measures the grid in row height.

Note: You can also use `setDensityMapMode` to set the threshold, and specify whether the map displays bins in microns or rows.

Use this command any time after placement.

Use the Display Density Map form to select colors and step-size for the density levels in the map.

Parameters

`-gridInMicron` *microns*

Specifies the horizontal or vertical dimension of a bin in the map, measured in microns.

Default: `50`
Upper limit: `1000`

`-gridInRow` *numberRows*

Specifies the horizontal or vertical dimension of a grid in the map, measured in row height. If you specify a value less than `1`, the software uses 1.

Default: `10`
Upper limit: `100`

`-help`

Outputs a brief description that includes type and default information for each `reportDensityMap` parameter.

For a detailed description of the command and all of its parameters, use the man command: `man reportDensityMap`.

`-threshold` *density*

Specifies a threshold density value for the grid. Any grid under the threshold value is not reported. The value must be positive.

Default: `0.75`

Related Topic

- "Place Menu" chapter of the *Encounter Menu Reference*
  - Display Density Map

reportScanCell

`reportScanCell`

Displays a list of the scan cells that are read from the timing library or by the `specifyScanCell` command.

Parameters

None

Example

Following is an example of the output from `reportScanCell`:

```
ScanFF SEDFFXL
  FT SI is test_scan_in
  FT SE is Scan Enable Clock
  FT QN is test_scan_out_inverted
  FT Q is test_scan_out
ScanFF SEDFFX4
  FT SI is test_scan_in
  FT SE is Scan Enable Clock
  FT QN is test_scan_out_inverted
  FT Q is test_scan_out
...
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

reportScanChainPartition

```
reportScanChainPartition
```

Generates a report that lists compatible scan chains in scan chain partitions.

Specify scan chain partitions with the following command: specifyScanChainPartition.

Parameters

None

Example

To see a list of the compatible scan chain groups in a design with one partition, `clk1Domain`, that has two compatible scan chains, type the following command:

```
reportScanChainPartition
```

The command outputs the following messages:

```
Info: Scan Chain Partition Group set to:
      Partition group: clk1Domain
                chain: chain1
                chain: chain2
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

reportSiteUtilization

```
reportSiteUtilization
     [-area x1 y1 x2 y2]
     [-site list_of_site_names]
```

Reports the cell density of individual sites in the design, a specified area, or for a list of sites. Does not report the density of row sites covered by placement blockages or overlapping row sites.

Parameters

`-area` *`x1 y1 x2 y2`*

Specifies the coordinates of the area for which to report cell density.

`x1` Specifies the area's lower left x coordinate.

`y1` Specifies the area's lower left y coordinate.

`x2` Specifies the area's upper right x coordinate.

`y2` Specifies the area's upper right y coordinate.

`-site` *`list_of_site_names`*

Specifies the list of sites for which to report utilization.

restoreClustering

```
restoreClustering [fileName]
```

Restores the clustering information and generates a clustered netlist. Use this command after generating a clustered netlist and saving the cluster information to a file.

Note: A clustered netlist uses the same logical hierarchy as the original netlist.

Parameters

fileName	Specifies the filename for the cluster information. *Default:* If you do not specify this parameter, it is *cellName*.clu.gz.

restorePlace

`restorePlace` *`fileName`*

Loads the placement file. The placement file must be in Encounter format.

Use this command after loading the floorplan file that was used to generate the placement file.

Parameters

`fileName`	Specifies the placement filename.

Example

The following command loads the placement file `TOPCHIP_SP.place`:

```
restorePlace TOPCHIP_SP.place
```

savePlace

`savePlace` *`fileName`*

Writes the placement information to a file.

Parameters

`fileName`	Specifies the name of the placement information file.

Example

The following command writes the placement data to the file `mydesign.place`:

```
savePlace mydesign.place
```

scanReorder

```
scanReorder
     [-allowSwapping {true | false}]
     [-clkAware {true | false}]
     [-keepPDPorts {true | false}]
     [-keepPort filename]
     [-preferH {true | false}]
     [-preferV {true | false}]
     [-scanEffort {low | medium | high}]
     [-skipMode {skipNone | skipBuffer | skipTwoPinCell}]
```

Reorders the scan cells after running placement. Controls the skipping and removal of buffers and inverters in the scan chain.

Set global parameters for this command with the `setScanReorderMode` command.

This command automatically detects multiple power domains and minimizes the number of scan chains that pass through them. It respects rectilinear and rectangular power domain boundaries. If a scan chain crosses power domains that operate at the same voltage, the software does not insert level shifters. If the scan chain crosses power domains operating at different voltages, the software inserts the appropriate level shifters.

If scan reorder fails, use the `scanTrace -verbose` command to see where the scan chain stops.

Note: Parameters set by this command override the settings of the `setScanReorderMode` command.

Parameters

`-allowSwapping {true | false}`

Allows the software to swap scan elements between scan chains within the same partition.
Default: `false`

`-clkAware {true | false}`

Specifies whether scan reordering is clock tree aware.

- When `true`, scan reordering is clock tree aware, and is clock tree driven.
- When `false`, scan reordering is not clock tree aware, and is wirelength driven.

Default: `false` (The software is not clock tree aware.)

`-keepPDPorts {true | false}`

Maintains or corrects the hierarchical ports without deleting or inserting shifters, and reorders the remaining scan chain.

To turn this option off, use `scanReorder -keepPDports false` or `setScanReorderMode -keepPDPorts false`.

Default: `true`

Note: This parameter works in the MSV flow only.

`-keepPort` *`fileName`*

Specifies a file that lists the names of hierarchical instances whose ports cannot be changed, removed, duplicated, or reordered during scan reordering.
Default: `""`

`-preferH {true | false}`

Specifies that the preferred direction for scan reordering connections is horizontal. Reorders the scan chain to snake a sideways "S" configuration if the block is short and wide or the design has vertical routing congestion.
Default: `false`

Note: If both `-preferV` and `-preferH` are set to `true`, the software does not honor the setting and treats them both as if they were `false`.

`-preferV {true | false}`

Specifies that the preferred direction for scan reordering connections is vertical. Reorders the scan chain to snake a vertical "S" configuration if the block is long and narrow or the design has horizontal routing congestion.
Default: `false`

Note: If both `-preferV` and `-preferH` are set to `true`, the software does not honor the setting and treats them both as if they were `false`.

`-scanEffort`

Specifies the effort level for reordering scan chains.
Default: `-medium`

Specify one of the following values:

`-high`

Runs more iterations of scan reorder to achieve better net length. Using this parameter increases the run time.

`-low`

Runs fewer iterations of scan reorder to minimize net length quickly.

`-medium`

Runs a normal number of iterations.

`-skipMode {-skipNone | -skipBuffer | -skipTwoPinCell}`

Specifies how scan reordering handles buffers and inverters in the scan chain.
Default: `-skipNone`

Specify one of the following:

`-skipNone`

Specifies that buffers and inverters remain after the scan chain reorder.

`-skipBuffer`

Ignores buffers in the scan chain. If a buffer or inverter is not connected elsewhere in the design, `scanReorder` removes the buffer.

`-skipTwoPinCell`

Ignores buffers and inverters in the scan chain. If a buffer or inverter is not connected elsewhere in the design, `scanReorder` removes the buffer or inverter.

Command Order

Use this command after running placement with the ignore scan connection parameter (`setPlaceMode -ignoreScan true`), and after specifying the scan chain or loading scan chain information in the DEF or TDF files.

Example

The following command reorders the scan cells using the sequence in the DEF file as the initial order of scan elements, ignoring netlist mismatches:

```
scanReorder -defInForce
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

scanTrace

```
scanTrace
     [-lockup | -noLockup]
     [-verbose]
```

Traces the scan chain connections and reports the starting and ending scan points and the total number of elements in the scan chain. The `scanReorder` command automatically calls `scanTrace` internally if you have not previously run `scanTrace`.

Note: If scan chains are specified by reading in a DEF file, the software does a native scan trace. The scan DEF file is stored in the database and, when the `scanReorder` command runs, the software matches the scans and honors the ordered sections. However, if you specify setPlaceMode -reorderScan false, the software does not perform scan chain reordering, so the DEF file will not include the + `ORDERED` statement in the `SCANCHAINS` section.

Parameters

`-lockup`	Specifies that the tracing detects lockup latches automatically. **Note:** If the lockup elements are flops (non scan flip-flops), the lockup elements must be identified using the `specifyLockupElement` command prior to running `scanTrace` in its default mode.
`-noLockup`	Specifies that the tracing will not detect lockup elements. This is the default.
`-verbose`	Prints the cell instance names that `scanTrace` determines are part of the scan chain. If scan chain tracing fails, use this parameter to see where the scan chain tracing stops.

Command Order

Use this command after specifying scan chains using the `specifyScanChain` command.

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

setDensityMapMode

```
setDensityMapMode
     [-help]
     [-reset]
     [-gridInMicron microns | -gridInRow numberRows]
     [-threshold density]
```

Sets global parameters for the `reportDensityMap` command. Parameters that you specify with `setDensityMapMode` are then used when you run `reportDensityMap`.

Note: Parameters specified by the `reportDensityMap` command supersede those specified by the `setDensityMapMode` command.

See `getDensityMapMode` to return the current settings for `setDensityMapMode`.

Parameters

`-gridInMicron` *`microns`*

Specifies the horizontal or vertical dimension of a grid in the map, measured in microns.

Default: `50`

`-gridInRows` *`numberRows`*

Specifies the horizontal or vertical dimension of a grid in the map, measured in number of rows. If you specify a value less than `1`, the software uses `1`.

Default: `10`

`-help`

Outputs a brief description that includes type and default information for each `setDensityMapMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command:
`man setDensityMapMode`.

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setDensityMapMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-threshold` *`density`*

Specifies a threshold density value for the grid. Any grid under the threshold value is not reported. The value must be positive. Cadence recommends an upper limit of `1.20`.

Default: `0.75`

Examples

- The following command resets the `-gridInRow` parameter to its default value:

  ```
  setDensityMapMode -reset -gridInRow
  ```

- The following command resets all `setDensityMapMode` parameters to their default values:

  ```
  setDensityMapMode -reset
  ```

setFillerMode

```
setFillerMode
     [-help]
     [-reset]
     [-core {listOfCoreFillerCells}]
     [-corePrefix prefix]
     [-createRows {true | false}]
     [-deleteFixed {true | false}]
     [-doDRC {true | false}]
     [-ecoMode {true | false}]
     [-fitGap {true | false}]
     [-honorPrerouteAsObs {true | false}]
     [-merge {true | false}]
     [-minHoleCheck {true | false}]
```

Controls certain aspects of how the software adds filler cells.

Use `getFillerMode` to return the current settings for the `setFillerMode` command.

The `setFillerMode` parameters affect the behavior of the following commands:

- `addFiller`
- `deleteFiller`

  Note: Parameters specified by the `addFiller` or `deleteFiller` command supersede those specified by the `setFillerMode` command.

Parameters

`-core` *`listOfCoreFillerCells`*

Specifies the filler cells to add with `addFiller`. Enclose the filler cell names in quotation marks (") or curly braces ({ }).
Default: `""` (empty string)

`-corePrefix` *`prefix`*

Specifies the prefix for the added filler cells.
Default: `FILLER`

`-createRows {true | false}`

Instructs `addFiller` to create rows in the floorplan.
Default: `true`. If a cell's site does not have rows in the floorplan, `addFiller` creates the rows.

`-deleteFixed {true | false}`

Deletes `FIXED` filler cells.
Default: `true`. End-cap and well-tap cells are marked `FIXED`, and `deleteFiller` can delete them.

`-doDRC {true | false}`

When `addFiller` is called after routing, instructs the software to run DRC checks of filler pins with signal net wires. To disable the DRC checks, set this parameter to `0`.
Default: `true`

`-ecoMode {true | false}`

Resolves overlaps of standard cells with fillers by removing the overlapping fillers and inserting new fillers in existing or newly created gaps due to partial overlaps. If you use this parameter after routing, the command also resolves DRC violations of filler geometries of all fillers with regular wires.
Default: `false`

`-fitGap {true | false}`

Fills a gap between cells by adding a combination of cells, instead of by adding the single largest cell that fits, if doing so avoids leaving an unfilled single-width gap.
Default: `false`

`-help`

Outputs a brief description that includes type and default information for each `setFillerMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setFillerMode`.

`-honorPrerouteAsObs {true | false}`

Instructs `addFiller` to honor the setting of the setPrerouteAsObs command. By default, `addFiller` does not consider any preroutes to be obstructions.

When this parameter is `true`, `addFiller` treats preroutes on metal layers where filler cells have geometries as obstructions, and ignores preroutes on other layers. In most cases, this means that `addFiller` treats preroutes on *metal1* as obstructions and ignores preroutes on other metal layers. For example, if `setPrerouteAsObs` is set to *metal1* and *metal2*, and all the filler cell geometries are on *metal1*, `addFiller` treats *metal1* preroutes as obstructions but does not treat *metal2* preroutes as obstructions.
Default: `false`. `addFiller` ignores `setPrerouteAsObs` and inserts fillers in all legal sites.

`-merge {true | false}`

When checking for spacing violations, merges same-net geometries to determine width-dependent spacing. If the LEF file contains width-dependent rules, this parameter causes `addFiller` to have the following behavior:

- If you specify `-merge true`, the command merges overlapping same-net objects to create a large rectangle.
- If you do specify `-merge false`, the command uses the shape of the object from the database.

Default: `false`

Tip

To avoid spacing violations, Cadence recommends that you run `addFiller` or `setFillerMode` with the `-merge true` parameter if you run `verifyGeometry` with the `-merge` parameter specified.

`-minHoleCheck {true | false}`

Calls `verifyGeometry` to check for minimum hole violations before adding filler cells and repairs the violations when `addFiller` runs.
Default: `false`

`-reset`	Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setFillerMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

Example

- The following command assigns a prefix of ABC to filler instances of cells `FILL64` and `FILL8` when `addFiller` inserts the instances:

  ```
  setFillerMode -corePrefix ABC -core "FILL64 FILL8"
  ```

- The following command resets the `-minHoleCheck` parameter to its default value:

  ```
  setFillerMode -reset -minHoleCheck
  ```

- The following command resets all `setFillerMode` parameters to their default values:

  ```
  setFillerMode -reset
  ```

setPlaceMode

```
setPlaceMode
     [-help]
     [-reset]
     [-blockedShifterCols {true | false}]
     [-blockedShifterRows {true | false}]
     [-checkPinLayerForAccess {list_of_layer_numbers}]
     [-checkRoute {true | false}]
     [-clkGateAware {true | false}]
     [-colShiftersOnly {true | false}]
     [-congEffort {high | medium | low}]
     [-dividedShifterCols {true | false}]
     [-dividedShifterRows {true | false}]
     [-doCongOpt {true | false}]
     [-doRPlace {true | false}]
     [-fixedShifter {true | false}]
     [-fp {true | false}]
     [-hardFence {true | false}]
     [-honorImplantSpacing {true | false}]
     [-honorSoftBlockage {true | false}]
     [-ignoreScan {true | false}]
     [-ignoreSpare {true | false}]
     [-incrPlaceForOpt {true | false | auto}]
     [-maxDensity value]
     [-maxRouteLayer layer_number]
     [-maxShifterColDepth value]
     [-maxShifterDepth value]
     [-maxShifterRowDepth value]
     [-moduleAwareSpare {true | false}]
     [-modulePadding module factor]
     [-modulePlan {true | false}]
     [-padFixedInsts {true | false}]
     [-padForPinNearBorder {true | false}]
     [-placeIoPins {true | false}]
     [-powerDriven {true | false}]
     [-preserveRouting {true | false}]
     [-reorderScan {true | false}]
     [-rmAffectedRouting {true | false}]
     [-rowShiftersOnly {true | false}]
     [-strictShifterSide {true|false}]
     [-strictShifterSpot {true|false}]
     [-swapEEQ {true | false}]
     [-tdInstPadding {true | false}]
     [-timingDriven {true | false}]
     [-viaInPin {true | false}]
     [-wireLenOptEffort {none | medium | high}]
```

Controls certain aspects of how the software places cells.

Use the getPlaceMode command to return the current settings for the setPlaceMode command.

The setPlaceMode parameters affect the behavior of the following commands:

- placeDesign
- refinePlace

  Note: Parameters specified by the placeDesign or refinePlace commands supersede those specified by the setPlaceMode command.

Parameters

-blockedShifterCols {true | false}

Prevents standard cells from being placed near shifters on vertical edges of the power domain boundary.
Default: false

-blockedShifterRows {true | false}

Prevents standard cells from being placed near shifters on horizontal edges of the power domain boundary.
Default: false

-checkPinLayerForAccess {*list_of_layer_numbers*}

Specifies the layers to check for pin access.

If the design has pins on *metal1* and *metal2*, to check pin access for pins on both layers, type the following command:

```
setPlaceMode -checkPinLayerForAccess {1 2}
```

This command checks pin access for *metal1* pins from *metal2* and for metal2 pins from metal3.

Default: By default, a pin's access check is done only for pins on *metal1*.

`-checkRoute {true | false}`

Considers pre-routed `FIXED` signal wires and avoids creating DRC violations between `FIXED` signal wires and instance pins. When this parameter is not specified, `refinePlace` ignores `FIXED` signal wires.
Default: `false`

Note: Using this parameter might result in longer run time.

If you specify `refinePlace -ioPinBlockage`, you do not need to specify `-checkRoute` for signal pins because `refinePlace` will treat the signal pins similar to supply pins for blockages. However, signal wires still need this parameter.

`-clkGateAware {true | false}`

Specifies that placement is aware of clock gate cells in the design.
Default: `false`

Note: The `setOptMode` command also has a `-clkGateAware` parameter. For best results, if you set `setPlaceMode -clkGateAware` to `true`, set `setOptMode -clkGateAware` to `true` also.

`-colShiftersOnly {true | false}`

Prevents shifters from being assigned to horizontal sides.
Default: `false`

`-congEffort {high | medium | low}`

Specifies the effort level for relieving congestion.
Default: `medium`

Specify one of the following values:

`high`	Runs more iterations of placement in an effort to achieve better congestion results. This parameter increases the run time.
`low`	Runs fewer iterations of placement to arrive quickly at a legal placement. This parameter might decrease placement quality.
`medium`	Runs placement on designs at a normal effort level.

`-dividedShifterCols {true | false}`

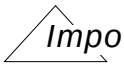
Forces the level shifters assigned to vertical edges of a power domain boundary to seek out the power domain corners. Otherwise, the shifters can be placed anywhere on those edges, usually trying to minimize wire length.
Default: `false`

`-dividedShifterRows {true | false}`

Forces the level shifters assigned to horizontal edges of a power domain boundary to seek out the power domain corners. Otherwise, the shifters can be placed anywhere on those edges, usually trying to minimize wire length.
Default: `true`

`-doCongOpt {true | false}`

Moves cells to improve wirelength and congestion during detailed placement. This parameter is honored only when `-modulePlan` is `false`.
Default: `false`



Important

This parameter is obsolete and will be removed in a future release. Cadence recommends that you use `-congEffort high` instead of this parameter to relieve congestion.

`-doRPlace {true | false}`

Specifies whether to run detailed placement (`refinePlace`) at the end of `placeDesign`, to legalize placed instances.
Default: `true`

`-fixedShifter {true | false}`

Specifies whether to set the shifters to `FIXED` status after placing them.
Default: `false`

`-fp {true | false}` Runs placement in floorplan mode. This mode is used for prototyping and runs quickly to gauge the feasibility of the netlist, but might not place design components in legal locations.
Default: `false`

`-hardFence {true | false}]`

Specifies whether `refinePlace` and honors or ignores the fence and region placement constraints while legalizing placed instances.
Default: `true` (`refinePlace` honors the constraints)

`-help`

Outputs a brief description that includes type and default information for each `setPlaceMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setPlaceMode`.

`-honorImplantSpacing {true | false}`

Makes the `refinePlace` command aware of implant layers. Specifying this parameter allows you to enforce minimum spacing rules between low voltage threshold (LVT) and high voltage threshold (HVT) cells. The spacing rules are enforced for cells in the same row.

To enable this parameter, complete the following steps:

- Specify the following in the LEF file:
  - Include implant layer geometries in the macro obstruction statements for standard cells.
  - Include a value for the minimum spacing between implant layers in the Layer (Implant) statement.

  For more information on LEF, see "LEF Syntax" in the *LEF/DEF Language Reference*.
- Insert filler cells between the LVT and HVT cells to provide proper spacing.

Default: `false`

`-honorSoftBlockage {true | false}`

Specifies that soft blockages will be considered when running detailed placement. By default, soft placement blockages are considered during global placement only, and are ignored during `refinePlace` and placement optimization.
Default: `false`

`-ignoreScan {true | false}`

Disregards scan connections while placing scan groups. Before you place the design, you must already have specified the scan cells with the `specifyScanCell` command, or the scan cell information must be in the timing library.
Default: `true`

Note: Scan groups are sometimes called scan partitions.

`-ignoreSpare {true | false}`

Disregards spare cell connections while placing spare cells. Before you place the design, you must already have specified the spare cells with the `specifySpareGate` command, or the spare cell information must be in the timing library.
Default: `true`. `placeDesign` ignores spare cell connections.

Note: The placer disconnects high-fanout (nets with more than 75 terminals) from spare cells, even when this parameter is set to `false`.

`-incrPlaceForOpt {true | false | auto}`

Integrates with `refinePlace` to resolve the local hot spot areas and make the timing improvement more stable during `optDesign`.
Default: `auto`

`-maxDensity` *`value`*

Controls local density during global placement. Sets the maximum placement density of the core area so that the placement engine aggressively minimizes the wirelength, while satisfying the maximum density constraint. This helps to achieve better timing results in low utilization designs. After global placement, the density control is removed.

Use a value between 0 (zero percent) and 1 (100 percent). For example, a value of 0.5 indicates that the maximum utilization of the entire design will be 50 percent or less.
Default: `-1.000`

`-maxRouteLayer` *`layer_number`*

Constrains placement up to the specified layer for congestion and routability estimation.
Default: `""` (empty string)

`-maxShifterColDepth` *`microns`*

Specifies the maximum distance from a vertical side that a shifter can be placed.
Default: `999` (no effect)

`-maxShifterDepth` *`microns`*

Specifies the maximum distance from a horizontal or vertical side that a shifter can be placed.
Default: `999` (no effect)

`-maxShifterRowDepth` *`microns`*

Specifies the maximum distance from a horizontal side that a shifter can be placed. The distance is measured from the power domain boundary, not from the edge of the placeable areas and accounts for gaps.
Default: `999` (no effect)

`-moduleAwareSpare {true | false}`

Places spare cells in the netlist within a hierarchical module, even if no region or fence constraints are specified.

- When this parameter is `true`, spare cells in a logical hierarchy are placed within the bounds of the hierarchy.
- When this parameter is `false`, spare cells are spread out within the placement area, and are not bound to a hierarchical module.

Default: `false`

`-modulePadding` *`module factor`*

Specifies a module in which to add padding (placement clearance) and a factor to use to calculate the padding dimension. Adding padding reduces placement density and localized congestion (hotspots) by spreading out cell instances in the specified modules. Module padding provides guidance for global placement only, and is ignored during placement legalization (`refinePlace`).

Default: `""` (empty string)

Note: This parameter is not enabled when `-modulePlan false` is specified.

module	Specifies a hierarchical instance to which the factor applies. You can use wildcards when you specify the instance name.
factor	Specifies a factor to use to calculate the padding dimension. The placer multiplies the instance area of all cell instances under the specified module by the factor. For example, a factor of 2 means that the placer “sees” each cell as twice its actual size. In most cases, a factor of 1.2 (which increases the area by 20 percent) is good enough. The placer ignores factors that are less than 1.0. *Default:* 1

`-modulePlan {true | false}`

Improves wire length, timing, relative module location, and congestion results during the early phase of global placement for most designs. In default mode (when `-modulePlan` is `true`) make sure you pre-place hard blocks (that is, set them to `FIXED` status) before running placement. If you do not, the command sets them to `FIXED` without placing them.

`-modulePlan` has the following features:

- It supports multi-threading. If you specify `false` for this parameter, multi-threading is disabled during placement.
- It is compatible with full and incremental placement, prototyping mode, and the in-place optimization flow.
- It supports fences, regions, and guides.

  Consider removing legacy guides when `-modulePlan` is specified: As `-modulePlan` works to produce the best possible relative module location for a design, these guides may no longer be required for optimum timing results.
- It supports spare cells distribution, MSV flow, and is clock-gate aware.

Default: `true` (In the blackblob flow, this parameter is disabled.)

Note: Cadence recommends using specifying `false` for this parameter for designs with more than 1,000 floorplan constraints or other types of complex floorplan constraints.

`-padFixedInsts {true | false}`

Specifies whether the `refinePlace` command honors cell padding on `FIXED` instances.
Default: `false`

`-padForPinNearBorder {true | false}`

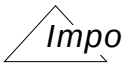
Adds padding to a cell if any signal pin is near enough to the border to cause DRC violations with any metal geometry.
Default: `false`

`-placeIoPins {true | false}`

Moves placed and unplaced I/O pins, based on the placement of connected instances in an attempt to find a better I/O pin placement than the one specified in the I/O pin placement or floorplan file. At the end of global placement, the I/O pin location is legalized.

When the value of this parameter is `false`, I/O pins are ignored during global placement and no legalization is done at the end.

Default: `true`



If `placeDesign` does not place the standard cells, for example if all instances are fixed or if there is no placeable area, then `placeDesign` also skips I/O pin assignment.

`-powerDriven {true | false}`

Identifies and constrains power-critical nets to reduce switching power. In most cases, timing is not degraded. However, in some cases, a trade-off between power and timing is required. An activity file (TCF or VCD) is required to supply the net switching information. Use the `read_activity_file` command to load the activity file.
Default: `false`

Note: In prototyping mode, power-driven placement is disabled.

`-preserveRouting {true | false}`

Preserves all routed wires.
Default: `false`. `refinePlace` deletes all routed wires.

`-reorderScan {true | false}`

Ignores scan chain connectivity during placement and performs scan chain reordering after placement.

If you specify `false` for this parameter, the software considers scan chain connectivity during placement but does not perform scan chain reordering after placement.

Default: `true`

`-reset`
: Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setPlaceMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-rmAffectRouting {true | false}`
: Specifies whether `refinePlace` removes wires connected to moved cells only or to all cells.
Default: `false`. Removes all wires, whether connected cells were moved or not.

`-rowShiftersOnly {true | false}`
: Prevents shifters from being assigned to vertical sides.
Default: `false`.

`-strictShifterSide {true | false}`
: Forces shifters to the side closes to the external connection.
Default: `false`

`-strictShifterSpot {true | false}`
: Forces shifters to the spot closest to the external connection.
Default: `false`

`-swapEEQ {true | false}`
: Determines whether master cells can be replaced by EEQ cells during detailed placement to improve routability. After detailed placement, the software reports the number of replacements. EEQ cells are defined in the LEF file.
Default: `false`

`-tdInstPadding {true | false}`
: Specifies whether `placeDesign` adds extra padding on instances on timing-critical paths.
Default: `false`

`-timingDriven {true | false}`

Takes timing into account during placement to improve the placement of instances on timing critical paths and builds a timing graph before placing the design. Before using this parameter, make sure the timing constraints are loaded in the design.
Default: `true`

Note: This parameter applies to the global placement only.

`-viaInPin {true | false}`

Specifies that pin access checks assume that the router drops a via to a pin only if the via is fully contained inside the pin geometry.
Default: `false`

Specify this parameter only when the `-routeWithViaInPin` parameter is also specified for `setNanoRouteMode`. For information, see "setNanoRouteMode" on page 1280.

`-wireLenOptEffort {none | medium | high}`

Optimizes wire length by swapping cells. Optimization can reduce the total wire length up to two percent without degrading placement. In high-effort mode, it might add up to ten percent to the run time.
Default: `none`

Examples

- The following command checks pin access on *metal1* from *metal3*:

```
setPlaceMode -checkPinLayerForAccess {1 2 3}
placeDesign
checkPlace
```

- The following commands add padding to the cells in module ABC so that the placer sees them as 1.5 times (50 percent larger) than their actual area:

```
setPlaceMode -modulePadding ABC 1.5
placeDesign
```

- The following command resets the `-checkPinLayerForAccess` parameter to its default value:

```
setPlaceMode -reset -checkPinLayerForAccess
```

- The following command resets all `setPlaceMode` parameters to their default values:

```
setPlaceMode -reset
```

Related Topics

- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

setPrerouteAsObs

```
setPrerouteAsObs {1 2 3 4}
```

Treats routing blockage objects and wires with the DEF attribute `+ SHAPE STRIPE` on the specified layers as placement blockages so that placement commands do not place cells under them.

Use this command before running placement.

To query the current setting for this command, use the `dbGetPrerouteAsObs` database command.

Note: Preplaced instances are not moved by placement commands, even if they violate the specified blockage.

Note: Fillers are still added under preroute wires.

Parameters

`{1 2 3 4}` Specifies the metal layer number(s).

In most all cases you must include metal layer 1.

Default

The default value is set after the library is read in by the floorplan section, and depends on the number of routing layers in the design.

- In designs with one, two, or three routing layers, preroutes on all layers are treated as obstructions. For example, in a three-layer design, preroutes on M1 M2, and M3 are treated as obstructions by default.
- In designs with more than three routing layers, only M1 and M2 preroutes are treated as obstructions by default. For example, in a five-layer design, preroutes on M1 and M2 are treated as obstructions by default.

Example

The following command specifies that cells must not be placed under power or ground stripes or routing blockages that are on layer 1, 2, or 5. The placer treats them as placement blockages.

```
setPrerouteAsObs {1 2 5}
```

Related Topic

- "Placement Commands" chapter in the *Encounter User Guide*
  - Placement Treatment of Preroutes

setScanReorderMode

```
setScanReorderMode
     [-help]
     [-reset]
     [-allowSwapping {true | false}]
     [-clkAware {true | false}]
     [-keepPDPorts {true | false}]
     [-keepPort fileName]
     [-preferH {true | false}]
     [-preferV {true | false}]
     [-scanEffort {low | medium | high}]
     [-skipMode {skipNone | skipBuffer | skipTwoPinCell}]
```

Controls certain aspects of how the software reorders scan chains.

Use the getScanReorderMode command to return the current settings for the setScanReorderMode command.

Note: Parameters specified by the scanReorder command supersede those specified by the setScanReorderMode command.

Parameters

`-allowSwapping {true | false}`

Allows the software to swap scan elements between scan chains within the same partition.
Default: `false`

`-clkAware {true | false}`

Specifies whether scan reordering is clock tree aware.

- When `true`, scan reordering is clock tree aware, and is clock tree driven.
- When `false`, scan reordering is not clock tree aware, and is wirelength driven.

Default: `false` (The software is not clock tree aware.)

`-help`

Outputs a brief description that includes type and default information for each setScanReorderMode parameter.

For a detailed description of the command and all of its parameters, use the man command:
`man setScanReorderMode`.

`-keepPDPorts {true | false}`

Maintains or corrects the hierarchical ports without deleting or inserting shifters, and reorders the remaining scan chain.
Default: `true`

Note: This parameter works in the MSV flow only.

`-keepPort` *`fileName`*

Specifies a file that lists the names of hierarchical instances whose ports cannot be changed, removed, duplicated, or reordered during scan reordering.
Default: `""`

`-preferH {true | false}`

Specifies that the preferred direction for scan reordering connections is horizontal. Reorders the scan chain to snake a sideways "S" configuration if the block is short and wide or the design has vertical routing congestion.
Default: `false`

Note: If both `-preferV` and `-preferH` are set to `true`, the software does not honor the setting and treats them both as if they were `false`.

`-preferV {true | false}`

Specifies that the preferred direction for scan reordering connections is vertical. Reorders the scan chain to snake a vertical "S" configuration if the block is long and narrow or the design has horizontal routing congestion.
Default: `false`

Note: If both `-preferV` and `-preferH` are set to `true`, the software does not honor the setting and treats them both as if they were `false`.

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setScanReorderMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-scanEffort`
Specifies the effort level for reordering scan chains.
Default: `-medium`

Specify one of the following values:

`-high`

Runs more iterations of scan reorder to achieve better net length. Using this parameter increases the run time.

`-low`

Runs fewer iterations of scan reorder to minimize net length quickly.

`-medium`

Runs a normal number of iterations.

`-skipMode {-skipNone | -skipBuffer | -skipTwoPinCell}`
Specifies how scan reordering handles buffers and inverters in the scan chain.
Default: `-skipNone`

Specify one of the following:

`-skipNone`

Specifies that buffers and inverters remain after the scan chain reorder.

`-skipBuffer`

Ignores buffers in the scan chain. If a buffer or inverter is not connected elsewhere in the design, `scanReorder` removes the buffer.

`-skipTwoPinCell`

Ignores buffers and inverters in the scan chain. If a buffer or inverter is not connected elsewhere in the design, `scanReorder` removes the buffer or inverter.

Examples

- The following command resets the `-keepPDPorts` parameter to its default value:

```
setScanReorderMode -reset -keepPDPorts
```

- The following command resets all `setScanReorderMode` parameters to their default values:

  ```
  setScanReorderMode -reset
  ```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

setTieHiLoMode

```
setTieHiLoMode
     [-help]
     [-reset]
     [-cell "tieHighCellName tieLoCellName" | -cell tieHiLowCellName]
     [-createHierPort {true | false}]
     [-honorDontTouch {true | false}]
     [-maxDistance distanceValue]
     [-maxFanOut fanOutValue]
     [-prefix prefixName]
     [-reportHierPort {true | false}]
```

Controls certain aspects of how the software adds or deletes tie-high and tie-low cells.

See getTieHiLoMode to query current settings for the setTieHiLoMode command.

The parameters of the setTieHiLoMode command affect the behavior of the following commands:

- addTieHiLo
- deleteTieHiLo

  Note: Parameters specified by addTieHiLo and deleteTieHighLo supersede those specified by the setTieHiLoMode command.

Parameters

-cell *cellName*	Specifies the tie cell names to be used by the addTieHiLo command. You can specify a maximum of two tie-cells, where one cell must be a tie-high driver, and the other a tie-low driver. The two tie-cell names must be in quotation marks. *Default:* "" (empty string)
-createHierPort {true | false}	Allows the added tie cells to connect to tie pins across hierarchical boundaries if other constraints, such as maximum fanout and maximum distance, allow it.
-help	Outputs a brief description that includes type and default information for each setTieHiLoMode parameter. For a detailed description of the command and all of its parameters, use the man command: man setTieHiLoMode.

`-honorDontTouch {true | false}`

Allows both `addTieHiLo` and `deleteTieHiLo` commands to honor the `dont_touch` property set on instances, nets, cells, and modules. For instances that need to be tied off, the `dont_touch` property has no effect, because the operation is on the net connected to one of its terms.
Default: `false`

`-maxFanOut` *`fanOutValue`*

Specifies the number of tie-pins a tie-net can drive. A value of `0` implies no fanout constraint.
Default: `0`

`-maxDistance` *`distanceValue`*

Specifies the distance, in microns, between the tie-cell driver and the tie-pins. A value of `0` implies no distance constraint.
Default: `0`

`-prefix` *`prefixName`*

Specifies a prefix for the added tie-cell instances.

`-reportHierPort {true | false}`

When the value of `-createHierPort` is `true`, reports created port connections to a file named `tiehilo.rpt`.
Default: `false`

`-reset` Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setTieHiLoMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

Example

- The following command sets `TIEHI` and `TIELO` cells a maximum distance of 20 microns between the tie-cell pin and the tie-off input pins:

  ```
  setTieHiLoMode -maxFanout 10 -maxDistance 20 -cell "TIEHI TIELO"
  ```

- The following command resets the `-maxDistance` parameter to its default value:

  ```
  setTieHiLoMode -reset -maxDistance
  ```

- The following command resets all `setTieHiLoMode` parameters to their default values:

```
setTieHiLoMode -reset
```

specifyCellPad

```
specifyCellPad
     leafCellNames
     factor
```

Specifies leaf cells to which to add padding (placement clearance) and a factor to use to calculate the padding dimension. The placer adds the padding on the right sides of the specified cells during placement. The padding is retained during optimization and CTS.

If there are congestion problems, delete the padding by using the `deleteAllCellPad` command. For information, see "deleteAllCellPad" on page 715.

Parameters

leafCellNames	Specifies the cell name(s) of a cell type to which the factor applies. You can use wildcards (*?) with this parameter.
factor	Specifies a factor to use to calculate the padding dimension. The placer multiplies the *metal2* pitch by the factor. It reads the *metal2* pitch value from the technology file.

Example

The following command adds padding that is twice the *metal2* pitch to leaf cells named `FJK2`:

```
specifyCellPad FJK2 2
```

Related Topic

- "Placing the Design" chapter in the *Encounter User Guide*
  - Adding Padding

specifyInstPad

`specifyInstPad` *`instanceName integer`*

Specifies a value for padding for an instance. Padding expands the area of an instance, so that the software leaves extra space around the instance during global placement (`placeDesign`), which helps reduce congestion in both the horizontal and vertical directions. The `refinePlace` and `checkPlace` commands ignore instance padding.

Adding instance padding reduces horizontal and vertical congestion, but also decreases utilization.

Note: To remove instance padding, use the `deleteInstPad` command.

Parameters

`instanceName`	Specifies the instance to pad. You can use wildcards (* or ?) when you specify the instance.
`integer`	Specifies the amount of padding to add, in terms of number of sites occupied by the instance (sites are defined in the LEF file). For example, if an instance occupies eight sites, specifying 8 expands the area of the instance by 100 percent, and specifying 12 expands the area of the instance by 150 percent. The number you specify must be a whole number that is equal to or greater than zero.

Related Topic

- "Placing the Design" chapter in the *Encounter User Guide*
  - Adding Padding
- "LEF Syntax" chapter in the Encounter *LEF/DEF Language Reference*
  - Site

specifyJtag

```
specifyJtag
     {-cell leafCellName
          | -inst instName
          | -hinst hInstName
          | -group {hInstName | cellName}
          | -groupWArea {hInstName | cellName width height} }
```

Specifies instances to place with the placeJtag command.

Parameters

`-cell` *`leafCellName`*

Marks all instances of the specified cell type as JTAG instances.

`-group` *`hInstName | cellName`*

Marks all hierarchical instances or intermediate instances as JTAG instances.

`-groupWArea` *`hInstName | cellName width height`*

Marks all instances inside the hierarchical instance or intermediate cells as JTAG instances, and specifies a rectangular width and height where all instances or cells are placed within that rectangular region.

`-hinst` *`hInstName`*

Marks a hierarchical instance as a JTAG instance.

`-inst` *`instName`*

Marks an instance as a JTAG instance. You can use a wildcard (*) character for this parameter.

Command Order

Use this command before running the `placeJtag` command.

Example

- The following command specifies that the cell `jtag_FlipFlop` will be placed when using the placeJtag command:

```
specifyJtag -cell jtag_FlipFlop
```

- The following command specifies that all instance names that begin with `instG` will be placed when using the `placeJtag` command:

```
specifyJtag -inst instG*
```

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os
- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

specifyLockupElement

```
specifyLockupElement
     {-cell leafCellName | -inst instName}
     -in ftname
     -out ftname
```

Specifies a cell or instance as a lockup element. Lockup elements are present in scan chains where the chain crosses between two different clock domains or clock phases. Use this command after running placement.

Parameters

`-cell` *`leafCellName`*
: Specifies the cell to be a lockup element.

`-in` *`ftname`*
: Specifies the name of the scan input.

`-inst` *`instName`*
: Specifies the instance to be a lockup element.

`-out` *`ftname`*
: Specifies the name of the scan output.

Example

The following command specifies that cell `TLATX1` is a lockup element with a scan input pin `D` and an output pin `Q`:

```
specifyLockupElement -cell TLATX1 -in D -out Q
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains

specifyScanCell

```
specifyScanCell
     cellName
     [-in ftname]
     [-out ftname]
     [-scanClock ftname]
     [-scanEnable ftname]
```

Specifies scan cells that are not listed in the timing library.

Parameters

cellName	Specifies name of the scan cell.
-in *ftname*	Specifies the name of the scan input.
-out *ftname*	Specifies the name of the scan output.
-scanClock *ftname*	Specifies the name of the scan clock.
-scanEnable *ftname*	Specifies name of the scan enable f-term or pin names.

Example

The following command specifies cell SDFFQ to be a scan cell:

```
specifyScanCell SDFFQ -in SI -out Q
```

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains
- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

specifyScanChain

```
specifyScanChain
     scanChainName
     -start {ftname | instPinName}
     -stop {ftname | instPinName}
```

Specifies a scan chain or group in a design. The actual tracing of the scan chain is performed by the scanTrace or scanReorder command. If scan chain specification fails, use the `scanTrace -verbose` command to see where the scan chain stops.

Note: Scan groups are sometimes called scan partitions.

Important

This command is not necessary for scan chains that are already specified in DEF or TDF files.

Parameters

`scanChainName`

Specifies a scan chain or scan group in a design.

`-start {`*`ftname`* `|` *`instPinName`*`}`

Specifies the starting scan pin name for the scan chain or scan group.

`ftname`	Specifies the design input pin name.
`instPinName`	Specifies the instance output pin name.

`-stop {`*`ftname`* `|` *`instPinName`*`}`

Specifies the stopping scan pin name for the scan chain or scan group.

`ftname`	Specifies the design output pin name.
`instPinName`	Specifies the instance input pin name.

Example

The following command sets scan chain `test_si` with a starting I/O pin name and a stopping pin name:

```
specifyScanChain test_si -start test23 -stop test_so
```

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - Optimizing and Reordering Scan Chains
- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

specifyScanChainPartition

```
specifyScanChainPartition
     -partition partitionName
     {-all | chainName1 chainName2 ...}
```

Defines a group (a partition) of compatible scan chains for the scan reordering flow. The software can swap elements in scan chains in the partition. Swapping elements helps reduce the total scan wire length in the partition and improves local congestion in the design.

To enable swapping, type the following command:

```
setScanReorderMode -allowSwapping true
```

Parameters

`-all` All scan chains in the specified partition are compatible.

`chainName1 chainName2` …

Only the specified scan chains in the specified partition are compatible.

`-partition` *`partitionName`*

Specifies a name for the partition and a list of the scan chains that belong to the partition. The scan flops in these chains are swappable.

Example

The following commands define two scan chains using `specifyScanChain` command, and define one partition using `specifyScanChainPartition`. The `setScanReorderMode -allowSwapping` parameter allows the software to swap elements between compatible chains.

```
specifyScanChain chain1 -start SI1 -stop SO1
specifyScanChain chain2 -start SI2 -stop SO2
specifyScanChainPartition -partition clk1Domain chain1 chain2
setScanReorderMode -allowSwapping true
scanReorder
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*

- Optimizing and Reordering Scan Chains

specifySpareGate

```
specifySpareGate
     {-cell leafCellName
        |-inst instanceName
        | -hinst {hierarchicalInstanceName | groupName}}
```

Specifies spare gates to place. To localize spare gate placement, guide the module containing the spare gate during the floorplanning session.

Parameters

`-cell leafCellName`

Specifies the cell type names of the spare gate cells. You can use this parameter to specify leaf cells (standard cells)—you cannot use it to specify spare modules. You can use wildcards with this parameter.

`-hinst {hierarchicalInstanceName | groupName}`

Specifies a hierarchical instance or group that contains spare gate cells. A hierarchical instance acts as one unit (a submodule). The units are evenly spaced and are close together when they are distributed.

This command does not recognize wildcards in hierarchical instance or group names.

`-inst instanceName`

Specifies the leaf instance names of the spare gates. You can use wildcards when specifying instance names.

Examples

- The following command specifies all cell types `YFD2S_SPG` to be spare gates:

  ```
  specifySpareGate -cell YFD2S_SPG
  ```

- The following command specifies instance `SH16/I1022` to be a spare gate:

  ```
  specifySpareGate -inst SH16/I1022
  ```

- The following command specifies that all instances in `SH16` to be spare gates:

  ```
  specifySpareGate -inst SH16/*
  ```

- The following command treats all instances with the cell type `NAND2X1` as spare cells. If the design uses some of these cells as spares and some as not spare, use the `-inst` parameter instead.

  ```
  specifySpareGate -cell NAND2X1
  ```

Related Topic

- "Placing the Design" chapter in the *Encounter User Guide*
  - Placing Spare Cells and Spare Modules
- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

traceJtag

```
traceJtag
     -infile jtagPinFile
     -outfile jtagInstanceFile
```

Traces JTAG logic between the I/O pins and the pins of given leaf instances. The input file contains a list of pins of instances that are the endpoints of JTAG logic. The command writes out a Tcl file that lists all the JTAG logic traced. You can then source the file to add this information to the Encounter database.

Parameters

`-infile` *`jtagPinFile`*

Specifies a file that lists the endpoints of JTAG logic. In the file, list each leaf instance or list the cell and its port.

- To list the leaf instances, use the `instpin` statement, as follows:

  `instpin` *`leafInstanceName pinName`*

- If all instances of a cell are used only in JTAG logic, you can list that cell and its port by using the `cellport` statement, as follows:

  `cellport` *`cellTypeName portName`*

  To avoid tracing non-JTAG cells, the cells you specify in a `cellport` statement must be JTAG-only cells. The cell type cannot be an intermediate cell. It must have at least one physical leaf instance.

`-outfile` *`jtagInstanceFile`*

Specifies the JTAG logic file in Tcl format.

Example

The following commands trace the JTAG logic based on the input file `jtag.leaf`, write the output file `jtag.out`, and add the logic specified in the file to the Encounter database:

```
traceJtag -infile jtag.leaf -outfile jtag.out
source jtag.out
```

Following is an example of an output file:

```
specifyJtag -inst regA
specifyJtag -inst regB
specifyJtag -inst regC
specifyJtag -inst regD
specifyJtag -inst regE
specifyJtag -inst regF
specifyJtag -inst regG
specifyJtag -inst regH
```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os

unplaceJtag

`unplaceJtag`

Undoes the placements made by the `placeJtag` command.

Parameters

None

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os

unspecifyJtag

```
unspecifyJtag
     {-cell leaf_cellName
      | -inst instanceName
      | -hinst hierarchicalInstanceName
      | -group hierarchicalInstanceName | cellName}
```

Unspecifies instances that were specified by the `specifyJtag` command.

Parameters

`-cell leaf_cellName`

Unmarks all instances of a specified cell type as JTAG instances.

`-group hierarchicalInstanceName | cellName`

Unmarks all hierarchical instances or intermediate instances as JTAG instances.

`-hinst hierarchicalInstanceName`

Unspecifies a hierarchical JTAG instance.

`-inst instanceName`

Unspecifies a JTAG instance. You can use wildcards (* or ?) with this parameter.

Example

- The following command unspecifies cell `jtag_FlipFlop` that was specified by the `specifyJtag` command:

  ```
  unspecifyJtag -cell jtag_FlipFlop
  ```

- The following command unspecifies all instance names that begin with `instG` that were specified by the `specifyJtag` command:

  ```
  unspecifyJtag -inst instG*
  ```

Related Topics

- "Placing the Design" chapter in the *Encounter User Guide*
  - Specifying and Placing JTAG and Other Cells Close to the I/Os

21

Power Analysis Commands

Note: The majority of these commands are obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

addPadLocation

`addPadLocation` *`x y`* [*`layerName`* [*`padInstName`* | *`voltage`*]]

Adds the location of a new power pad to memory.

Parameters

`layerName`	Specifies the generic metal routing layer name. Use a value such as `M1` or `V12`.
`padInstName`	Specifies the name of the power or ground pad instance at the design level.
`voltage`	Specifies the boundary voltage at the block level.
`x y`	Specifies the x and y coordinates of the power pad location.

Example

- The following command adds the location of a new power pad as x coordinate `787.277` and y coordinate `1290.035`:

  ```
  addPadLocation 787.277 1290.035
  ```

autoFetchDCSources

`autoFetchDCSources` *`netname`*

Returns a list of DC sources from your design. DC sources are identified in the LEF file as `CLASS PAD POWER`, or in the DEF file as `PIN` with `+USE POWER/GROUND`. You must save the list to a file, using the `savePadLocation` command.

Parameters

`netname`	Specifies the name of a power net for which to list DC sources. *Default:* If you do not specify a net name, the power analyzer automatically lists DC sources for all nets.

clearMacroISourceLocDisplay

`clearMacroISourceLocDisplay`

Clears the display of power current source locations for blocks. Note that the `I` in this command stands for *current*.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

None

clearPadLocDisplay

```
clearPadLocDisplay
     [-editForm]
```

Clears the display of the DC sources that were specified in the *Edit Pad Location* form.

Parameters

`-editForm`	Clears both the pad location list in the Edit Pad Location form and the display of DC sources that were specified on the form. If you do not specify this parameter, all DC source locations specified by the Display Pad Location form are cleared from the design display area.

clearRailAnalysisDisplay

```
clearRailAnalysisDisplay
```

Removes the graphically displayed results of rail analysis or VoltageStorm analysis from the design display area. Use this command after running power analysis.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

None

createMacroPinPGRails

```
createMacroPinPGRails
     [-cell cellName]
     [-instance instanceName]
```

Specifies the names of macro block instances or cells to be used by the updatePower command. The updatePower command creates virtual power and ground rails that connect feedthrough pins that are either horizontally or vertically aligned on specified macro block instances or cells. The software creates the virtual rails in both floorplan mode and layout mode. These created virtual rails are useful for providing connectivity information that would not otherwise be available for macro instances or cells. The createMacroPinPGRails command does not connect abutment pins, ring pins, pins for modules, or pin ports that are not aligned either horizontally or vertically.

Note: If you issue this command without parameters, the software automatically adds instances that contain a pin port shape of FEEDTHRU to the instance list, so that their pin port geometries are considered during rail analysis.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

-cell *cellName*
Specifies the name of the cell for which power and ground rails are to be created. Only instances that have a pin port shape of FEEDTHRU are added to the instance list. If you specify multiple cell names, separate each name with a space.

-instance *instanceName*
Specifies the name of the macro block instance for which power and ground rails are to be created. Only instances that have a pin port shape of FEEDTHRU are added to the instance list. If you specify multiple block names, separate each name with a space.

deletePadLocation

`deletePadLocation` *`x y`* [*`layerName`* [*`padInstName`* | *`voltage`*]]

Deletes the location of a power pad from memory.

Parameters

`layerName`	Specifies the generic metal routing layer name. Use a value such as `M1` or `V12`.
`padInstName`	Specifies the name of the power or ground pad instance at the design level.
`voltage`	Specifies the boundary voltage at the block level.
`x y`	Specifies the x and y coordinates of the power pad location.

Example

- The following command deletes the power pad location at x coordinate `787.277` and y coordinate `1290.035`:

  ```
  deletePadLocation 787.277 1290.035
  ```

displayMacroISourceLoc

`displayMacroISourceLoc` *`powerNetName`*

Displays the macro current source location for blocks. Note that the `I` in this command stands for *current*. The power pins of a block are both the power source for the block's contents, and the sinks for the power grid external to the macro. Use this command after running power analysis.

- For floorplan mode, the current source for blocks is displayed in a distributed manner for the grid the block is in. The grid is set up by the power structures.
- For layout mode, the current sources are displayed at the power pins of the blocks.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`powerNetName`	Specifies the name of power or ground net.

Example

- The following command displays the `vdd` current source for the blocks:

```
displayMacroISourceLoc vdd
```

displayPadLoc

`displayPadLoc -infile` *`filename`*

Loads and displays the pad location data from a file.

Parameters

`-infile` *`filename`*	Specifies the name of the file that contains pad location data.

displayRailAnalysisResults

```
displayRailAnalysisResults
     -net netName
     -type analysisType
     [-inDir dirPath]
     [-readInstancePowerFile instancePowerFileName]
     [-showUnconnectedInst]
     [-filter {v1 min1 max1} {v2 min2 max2} {v3 min3 max3} {v4 min4 max4}
          {v5 min5 max5} {v6 min6 max6} {v7 min7 max7} {v8 min8 max8}]
     [-visibleLayer layerName ...]
     [-emLimit {layerName * value} ...]
     [-violationBrowser]
```

Loads and displays the results of the Rail analysis or VoltageStorm analysis that was done using the updatePower or runVStorm command. Use this command after running power analysis using the Encounter® Power System (Next-Generation VoltageStorm) interface.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-emLimit {layerName * value}`

Specifies the EM limit for the layers for which you want to display rail analysis. For each layer, you must specify the layer name, an asterisk (which serves as a placeholder to indicate all widths), and a layer limit. You must enclose these three items in curly braces. You can specify EM limits for any number of metal layers and cut layers in the design.

If you do not specify this parameter, EM layer limits are not used.

`-filter {`*`v1 min1 max1`*`} {`*`v2 min2 max2`*`} {`*`v3 min3 max3`*`} {`*`v4 min4 max4`*`} {`*`v5 min5 max5`*`} {`*`v6 min6 max6`*`} {`*`v7 min7 max7`*`} {`*`v8 min8 max8`*`}`

Specifies the range for a filter as well as whether the values in that filter are to be displayed. You must specify all eight filter ranges in either increasing or decreasing order. These must be set as a triplet value enclosed within curly braces `{}`. The triplet consists of the following values

- *`v`* is the visibility control value. Specify 1 to make the values in this filter visible. Specify 0 to hide the values in this filter.
- *`min`* is a number that specifies the lower limit of the filter range.
- *`max`* is a number that specifies the upper limit of the filter range.

If you do not specify this parameter, the filter ranges defined in the results file are used for display.

`-inDir` *`dirPath`*

Specifies the path to a directory that contains the saved results.

Default: If you do not specify this parameter, the software searches for a file named $*`designName`*_$*`netName`*_vs2fe.$*`analysisType`*.

`-net` *`NetName`*

Specifies the name of the power net.

`-readInstancePowerFile` *`instancePowerFileName`*

Specifies the name of the instance power file for use in displaying either instance power or instance power density.

`-type` *`analysisType`*

Specifies the type of analysis results to be displayed.

Note: The *`analysisType`* values are not case-sensitive.

Analysis types are arranged to reflect the order in which they appear on the Display Rail Analysis Results form.

The values for the following three analysis types are derived from Encounter data in memory or from the saved results of an Encounter run.

`ird` IR drop (V).

`em` Electromigration (J/J*max*).

`ivd` Instance voltage drop (V).

Note: If you specify `type -ivd`, the `-visibleLayer` parameter is ignored.

The values for the following two analysis types are based on input instance power files that are independent of any particular software application:

`ip` Instance power (mW).

Provides a color gradient display of relative power distribution among a design's instances.

High power consumption by instances is indicated in the Encounter main window by colors at the red end of the spectrum.

Note: You must use the `-readInstancePowerFile` parameter if you intend to use this analysis type to display instance power colors in the Encounter main window.

`ipd` Instance power density (mW/μm^2).

Provides a color gradient display of relative power density among a design's instances.

Instance power density is obtained by dividing instance power by instance area.

High power density of instances is indicated in the Encounter main window by colors at the red end of the spectrum.

Note: You must use the `-readInstancePowerFile` parameter if you intend to use this analysis type to display instance power density colors in the Encounter main window.

The values for the following eight analysis types are derived from results in a VoltageStorm output directory:

`ir` IR drop (V).

`tc`	Tap current (A).
`rc`	Resistor current (A).
`rj`	Relative current density (J/J*max*).
`er`	Electromigration risk (1/h).
`vc`	Voltage source current (A).
`vv`	Voltages across vias and contacts (V).
`iv`	Instance-based supply voltage (V).

For more information about each of these types of analysis, see the *VoltageStorm PE Reference Manual*.

`-showUnconnectedInst`

Unconnected instances are shown in the color white in the design display area. This option is available only if you also specify `type -ivd`.

`-violationBrowser` Controls the display of certain rail analysis results in the Violation Browser.

`-visibleLayer` *`layerName`*

Specifies the names of the layers that are visible in the display. You can specify multiple layer names; but it is more effective to specify a single layer, to focus on the results for one layer at a time.

Note: If you specify `type -ivd`, the `-visibleLayer` parameter is ignored.

Default: If you do not specify this parameter, the display shows all layers.

Example

- The following command displays the IR drop results for the `vdd` net, contained in the results directory `VDD_25C_avg_1`:

  ```
  displayRailAnalysisResults -net vdd -type IR -inDir VDD_25C_avg_1
  -filter {1 1.799892 1.800000} {1 1.799784 1.799892} {1 1.799676 1.799784}
  {1 1.799568 1.799676} {1 1.799460 1.799568} {1 1.799352 1.799460}
  {1 1.799244 1.799352} {1 1.799136 1.799244} -visibleLayer M6 V56 M5 V45 M4 V34
  M3 V23 M2 V12 M1
  ```

  This example assumes that VoltageStorm was run using the `runVStorm` command and the `vdd` net was analyzed for IR. For a design named `dtmf_chip`, a file named `dtmf_chip_vdd_vs2fe.ir` was generated in the results directory after you issued the `runVStorm` command. That file is read when you issue the `displayRailAnalysisResults` command.

getLibraryPowerUnit

```
getLibraryPowerUnit libraryName
```

Reports the power unit value of the specified timing library file.
To set the power unit value of a timing library file, use the setLibraryPowerUnit command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Example

- The following command reports the power unit value of the `tim.lib` library:

  ```
  getLibraryPowerUnit tim.lib
  ```

getPGNetResis

```
getPGNetResis
     [-temperature degreesCelsius]
     pgNetName
     [-p2p x1 y1 layer1 x2 y2 layer2]
```

Computes the resistance of the power or ground net.

Parameters

`-p2p x1 y1 layer1 x2 y2 layer2`

Computes the effective resistance between two instance pins. The point/layer specification is used to find the corresponding pin for calculation.

- If you do not specify the option `-p2p`, the total resistance is the algebraic sum of the resistance of each individual wire and via.
- If you specify the option `-p2p`, the total resistance is the effective resistance of the entire resistor network.

Use the option `-p2p` with caution as it will take a while to get an effective R from large PG nets.

`pgNetName`

Specifies the name of the net for which resistance is to be computed. The command accepts multiple names.

Note: The total resistance is the algebraic sum of the resistance of each individual wire and via, rather than the effective resistance of the entire resistor network.

`-temperature degreesCelsius`

Specifies the temperature to be used during temperature-dependent resistance extraction. The temperature-dependent resistance coefficient is taken from the capacitance table.

Example

- The following command displays a report within the Encounter console:

  ```
  pgNetResis vdd!
  ```

  The report is generated with the format shown in the following example:

  ```
  Net Name: vdd!
  total resistance: 5.843993e+01 Ohm
      M1  resistance: 3.530900e+01 Ohm
      M2  resistance: 0.000000e+00 Ohm
      M3  resistance: 0.000000e+00 Ohm
      M4  resistance: 0.000000e+00 Ohm
      M5  resistance: 2.459333e+00 Ohm
      M6  resistance: 2.324933e+00 Ohm
      V12 resistance: 3.360000e+00 Ohm
      V23 resistance: 3.360000e+00 Ohm
      V34 resistance: 3.360000e+00 Ohm
      V45 resistance: 5.600000e+00 Ohm
      V56 resistance: 2.666667e+00 Ohm
    resistance of selected wires/vias: 1.666667e-01 Ohm
  ```

getPowerAnalysisLibrary

`getPowerAnalysisLibrary`

Displays the value set by the setPowerAnalysisLibrary command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

None

getPowerAnalysisSlew

```
getPowerAnalysisSlew
```

Displays the value set by the setPowerAnalysisSlew command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

None

getSpecialNetResis

```
getSpecialNetResis
     [-outfile fileName]
     [-temperature degreesCelsius]
     [-worst]
     [-p2p x1 y1 layer1 x2 y2 layer2]
     netName …
```

Reports resistance values for specified nets. Use this command after power planning.

Parameters

netName …	Specifies the names of regular or power/ground nets that appear in the SPECIALNETS section of the LEF file.
-outfile *fileName*	Creates a report containing the nets' resistance values. The power analysis software adds the extension .res to the *fileName* you specify.
-p2p *x1 y1 layer1 x2 y2 layer2*	Calculate resistance based on point-to-point coordinates and layer names.
-temperature *degreesCelsius*	Specifies the temperature to be used during temperature-dependent resistance extraction. The temperature-dependent resistance coefficient is taken from the capacitance table.
-worst	For cases where a PG net as three or more connections, reports information on the worst resistance on the net, and the path that has this worst resistance.

Example

- The following command creates a report called `report.res` containing resistance values for the nets VDD1 and VDD2:

```
getSpecialNetResis -outfile report VDD1 VDD2
Net Name: VDD1
  total resistance: 3.409200e+01 Ohm
    M1  resistance: 1.800000e-01 Ohm
    M2  resistance: 5.212000e+00 Ohm
    M3  resistance: 1.200000e+00 Ohm
    M4  resistance: 0.000000e+00 Ohm
    M5  resistance: 0.000000e+00 Ohm
    V12 resistance: 5.000000e+00 Ohm
    V23 resistance: 2.250000e+01 Ohm
    V34 resistance: 0.000000e+00 Ohm
    V45 resistance: 0.000000e+00 Ohm
  resistance of selected wires/vias: 0.000000e+00 Ohm

Net Name: VDD2
  total resistance: 3.409200e+01 Ohm
    M1  resistance: 1.800000e-01 Ohm
    M2  resistance: 5.212000e+00 Ohm
    M3  resistance: 1.200000e+00 Ohm
    M4  resistance: 0.000000e+00 Ohm
    M5  resistance: 0.000000e+00 Ohm
    V12 resistance: 5.000000e+00 Ohm
    V23 resistance: 2.250000e+01 Ohm
    V34 resistance: 0.000000e+00 Ohm
    V45 resistance: 0.000000e+00 Ohm
  resistance of selected wires/vias: 0.000000e+00 Ohm
```

- The following command reports the worst resistance for net SPNP_BUFF_REGC2_CVDD_T, which has more than two endpoints, and the path has the worst resistance.

```
getSpecialNetResis -worst SPNP_BUFF_REGC2_CVDD_T
####################################
Worst effective resistance on net SPNP_BUFF_REGC2_CVDD_T
between mtop/pll0 and mtop/sscg0 = 21.632 Ohms
####################################
```

- The following command calculates the resistance based on the point-to-point details (coordinates in microns and layer names).

```
getSpecialNetResis -p2p -5626 2185 M3 -5120 2550 M2 SPNP_BUFF_REGC2_CVDD_T
####################################
Effective resistance on net SPNP_BUFF_REGC2_CVDD_T
between points (-5626, 2185) and (-5120, 2550) = 4.6471 Ohms
####################################
```

loadPadLocation

```
loadPadLocation -file fileName
```

Loads a power pad location file to memory. Use this command after manually creating the power pad location file using the Edit Pad Location form, or using the savePadLocation command.

Parameters

fileName	Specifies the name of the power pad location file.

Example

- The following command loads the power pad location file `VDD.pp`:

```
loadPadLocation -file VDD.pp
```

probePower

```
probePower [hierInstName...]
```

Sets up the power probe for simulation-based power analysis. Issue the updatePower command to perform power analysis on the probed instances. The top-level module is probed by default. The average power waveform for the probed hierarchical instances is saved to the file *powerNetName*.fsdb. This file can be viewed by Debussy nWave, a third-party waveform viewer.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

hierInstName	Specifies the instance name or hierarchical instance name for which power is to be analyzed. The command accepts multiple names.

Example

- The following command sets the power probe for instances TOP/I412 and TOP/I413:

  ```
  probePower TOP/I412 TOP/I413
  ```

probePowerGraph

```
probePowerGraph
     -net netName
     {-node nodeId | -location x y -layer layerName}
```

Generates a report of wire and via rail analysis information in the log file.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-layer layerName`	The name of the layer for which the report is to be generated.
`-location x y`	The x and y coordinates of the node for which the report is to be generated.
`-net netname`	The name of the net for which the report is to be generated.
`-node nodeID`	The name of the node. If you specify this parameter, you do not need the `-location` and `-layer` parameters.

Example

- The following command generates a report for the instance on the `VDD` net on layer `m2` at the coordinates `2962, 3317`:

  ```
  probePowerGraph -net VDD -location 2962 3317 -layer m2
  ```

  The report is generated with the format shown in the following example:

  ```
  probing node 1618 on net "VDD":
           location: ( 2961.26 3310.08 ) um
           layer: M2
           north to south width: 58.36 um
           east to west width: 30.00 um
           V: 307.95 mV
           instance I: 0.00e+00 mA
           South neighbour:
              node: 1615
              location: ( 2961.26 3280.00 ) um
              layer: M2
              width: 30.00 um
              V: 307.21 mV
  ```

```
        I: 1.19e+01 mA
        R: 6.22e-02 ohm
     North neighbour:
    node: 1623
    location: ( 2976.07 3354.48 ) um
    layer: M2
    width: 0.39 um
    V: 309.04 mV
    I: -1.54e-01 mA
    R: 7.06e+00 ohm
North neighbour:
    node: 1622
    location: ( 2961.07 3354.48 ) um
    layer: M2
    width: 29.61 um
    V: 309.05 mV
    I: -1.18e+01 mA
    R: 9.30e-02 ohm
sum of current flowing in/out of node 1618: 4.67e-13 mA
```

reportVCDInvalidID

`reportVCDInvalidID` *`filename`*

Checks VCD files and generates a list of invalid ID nets. This is the list of nets that could not be mapped to an equivalent net from the Verilog® netlist, so the corresponding VCD activities cannot be asserted on these nets. The list can help in debugging the cause for missed VCD IDs.

Note: The list includes only invalid ID nets within the scope of the design top cell. Invalid ID nets in the VCD file that are outside scope of the design top cell are not included.

Use this command after extracting the RC data and before issuing the updatePower command. The list of invalid ID nets is written to the filename you specify with the `reportVCDInvalidID` command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`filename`	Specifies the name of the file to which the software writes the list of invalid ID nets.

Examples

- The following example shows the commands you would specify to generate an ASCII report of invalid IDs from the VCD file.

```
reportVCDInvalidID test.VCDinvalidID.log
updatePower
```

- The following example illustrates the messages that appear in the power analysis log file:

```
Total id in vcd file: 12345
    In module testbench/b1/cblk valid id: 1234
        redundant id: 123
    In module testbench/b1/cblk invalid id: 123
        redundant id: 12
```

  where:

  - `valid id` indicates nets that are defined in the VCD file and that were also found in the design.

- `invalid id` indicates nets in the VCD file that were not found in the design. Power analysis will not recognize these nets, resulting in no VCD switching activity for the nets.
- `redundant id` indicates nets in the VCD file that have the same reference ID in the VCD file. Valid IDs exist for such nets and power analysis recognizes VCD switching activity for the nets.

runVStorm

```
runVStorm
     {-dynamic | -static}
     [-analyzeDD {1 | 0}]
     [-analyzeDR {1 | 0}]
     [-analyzeER {1 | 0}]
     [-analyzeFD {1 | 0}]
     [-analyzeIR {1 | 0}]
     [-analyzeIV {1 | 0}]
     [-analyzePI {1 | 0}]
     [-analyzePV {1 | 0}]
     [-analyzeRC {1 | 0}]
     [-analyzeRJ {1 | 0}]
     [-analyzeTC {1 | 0}]
     [-analyzeVC {1 | 0}]
     [-analyzeVU {1 | 0}]
     [-analyzeVV {1 | 0}]
     [-biasVolt volts]
     [-cellPowerFile filename]
     [-cmd commandFile]
     [-currentFile {fileNames}]
     [-decapMethod {area | feasibility}]
     [-defaultFrequency megahertz]
     [-defVersion {5.4 | 5.5 | 5.6}]
     [-emLifetime value]
     [-emModelFile fileName]
     [-genBbv {1 | 0}]
     [-genIV {1 | 0}]
     [-genView {1 | 0}]
     [-genViewLef fileName]
     [-genViewView {detailed | abstract | reduced}]
     [-isPM {1 | 0}]
     [-ivFile fileName]
     [-keepTempFiles {1 | 0}]
     [-layerBias lefLayer value]...
     [-layerMap lefLayer icecapLayer]...
     [-libs dirNameList]
     [-msmvBiasVolt {nets}]
     [-msmvNet {nets}]
     [-net netName]
     [-netVolt volts]
     [-operationMode mode]
     [-outDir dirName]
     [-packMappingFile fileName]
     [-packModelFile fileName]
     [-pmDesignIncludeFile fileName]
     [-pmPowerIncludeFile fileName]
     [-pmPowerOutIncludeFile fileName]
     [-powerFile fileName]
     [-powerNetGroup {nets}]
```

```
[-powerUp {1 | 0}]
[-powerUpSpiceCorners corners]
[-powerUpSpiceModels fileNames]
[-powerUpSpiceNetlists fileNames]
[-powerUpSpicePWL " time1 voltage1 time2 voltage2..."]
[-ppFile fileName]
[-psoAOIR limit]
[-psoAONet net]
[-psoAOVolt voltage]
[-psoSFNet net]
[-psoSONet net]
[-runECO {1 | 0}]
[-runPM { 1| 0}]
[-simResolution nanoseconds]
[-simPeriod nanoseconds]
[-spef fileName]
[-tempDir dirName]
[-temperature degreesCelsius]
[-totalPower watts]
[-twf fileName]
[-useCellView type name view]...
[-vcdBased {1 | 0}]
[-vcdFile fileName]
[-vcdScope scopeName]
[-vcdStartTime nanoseconds]
[-vcdStopTime nanoseconds]
[-view value]
[-voltLimit volts]
[-vrangePercent value]
[-vsBeginIncludeFile fileName]
[-workDir dirName]
```

Enables you to access the VoltageStorm® PE power integrity verification solution, if you have the correct license and VoltageStorm® is in your path. Use this command after your design has been routed. Usually, however, you use it after you run Encounter power analysis. For more information about electromigration risk analysis, See the *VoltageStorm SoC Hierarchical PGS Manual*.

You can run the `runVStorm` command in one of two modes: `-static` or `-dynamic`. Each mode has its own set of parameters. Some of the parameters are common to both modes, but many parameters are unique to one mode or the other. Do not use `-static`

mode parameters with the `-dynamic` mode, and vice versa. The following table lists the mode support for each parameter.

runVStorm Parameters	Mode	
	-dynamic	**-static**
-analyzeDD {1 | 0}	**X**	
-analyzeDR {1 | 0}	**X**	
-analyzeER {1 | 0}	**X**	**X**
-analyzeFD {1 | 0}	**X**	
-analyzeIR {1 | 0}	**X**	**X**
-analyzeIV {1 | 0}	**X**	**X**
-analyzePI {1 | 0}	**X**	**X**
-analyzePV {1 | 0}	**X**	**X**
-analyzeRC {1 | 0}	**X**	**X**
-analyzeRJ {1 | 0}	**X**	**X**
-analyzeTC {1 | 0}	**X**	**X**
-analyzeVC {1 | 0}	**X**	**X**
-analyzeVU {1 | 0}	**X**	
-analyzeVV {1 | 0}	**X**	**X**
-biasVolt *value*	**X**	**X**
-cellPowerFile *fileName*		**X**
-cmd *commandFile*	**X**	
-currentFile **{***fileNames***}**	**X**	**X**
-decapMethod {area | feasibility}	**X**	
-defaultFrequency *megahertz*	**X**	
-defVersion {5.4 | 5.5 | 5.6}		**X**
-emLifetime *value*		**X**
-emModelFile *fileName*		**X**
-genBbv {0 | 1}		**X**

runVStorm Parameters	Mode	
	-dynamic	**-static**
-genIV {0 | 1}		**X**
-genView {0 | 1}		**X**
-genViewLef *fileName*		**X**
-genViewView {detailed | abstract | reduced}		**X**
-isPM {0 | 1}		**X**
-ivFile *fileName*	**X**	**X**
-keepTempFiles {0 | 1}		**X**
-layerBias *lefLayer value*		**X**
-layerMap *lefLayer icecapLayer*		**X**
-libs *dirNameList*	**X**	**X**
-msmvBiasVolt {*nets*}	**X**	**X**
-msmvNet {*nets*}	**X**	**X**
-net *netName*	**X**	**X**
-netVolt *voltage*	**X**	**X**
-operationMode *mode*	**X**	**X**
-outDir *dirName*		**X**
-packMappingFile *fileName*	**X**	
-packModelFile *fileName*	**X**	
-pmDesignIncludeFile *fileName*	**X**	
-pmPowerIncludeFile *fileName*	**X**	
-pmPowerOutIncludeFile *fileName*	**X**	
-powerFile *fileName*	**X**	**X**
-powerNetGroup {*nets*}	**X**	**X**
-powerUp {1 | 0}	**X**	
-powerUpSpiceCorners *corners*	**X**	
-powerUpSpiceModels *fileNames*	**X**	

runVStorm Parameters	Mode	
	-dynamic	**-static**
-powerUpSpiceNetlists *fileNames*	**X**	
-powerUpSpicePWL " *time1 voltage1 time2 voltage2...*"	**X**	
-ppFile *fileName*	**X**	**X**
-psoAOIR *limit*	**X**	**X**
-psoAONet *net*	**X**	**X**
-psoAOVolt *voltage*	**X**	**X**
-psoSFNet *net*	**X**	**X**
-psoSONet *net*	**X**	**X**
-runECO {0 | 1}	**X**	
-runPM {0 | 1}	**X**	
-simResolution *value*	**X**	
-simPeriod *value*	**X**	
-spef *fileName*	**X**	
-tempDir *dirName*		**X**
-temperature *degreesCelsius*	**X**	**X**
-totalPower *watts*		**X**
-twf *fileName*	**X**	
-useCellView {*type name view*}+		**X**
-vcdBased {0 | 1}	**X**	
-vcdFile *fileName*	**X**	
-vcdScope *scopeName*	**X**	
-vcdStartTime *nanoseconds*	**X**	
-vcdStopTime *nanoseconds*	**X**	
-view *value*	**X**	**X**
-voltLimit *voltage*	**X**	**X**
-vrangePercent *value*	**X**	**X**

runVStorm Parameters	Mode	
	-dynamic	**-static**
-vsBeginIncludeFile *fileName*		
-workDir *dirName*		**X**

The `runVStorm` command supports two methods of dynamic power calculation:

- Vectorless

  Vectorless dynamic power calculation does not require a VCD file. However, you must provide the following information, with which VoltageStorm calculates the required power for a VoltageStorm Dynamic Gate (VSDG) run:

  - Default frequency (in megahertz)
  - Resolution (in nanoseconds)
  - Simulation time (in nanoseconds)

- Vector-based

  Vector-based dynamic power calculation provides more accurate power results than the vectorless method. You must provide the following information for vector-based runs:

  - VCD file
  - Scope
  - Start time (in nanoseconds)
  - Stop time (in nanoseconds)
  - Default frequency (in megahertz)
  - Resolution (in nanoseconds)
  - Simulation time (in nanoseconds)

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-analyzeDD {1 | 0}`

Analyzes and reports on decoupling capacitance density.

`-analyzeDR {1 | 0}`

Analyzes and reports on decoupling capacitance required.

`-analyzeER {1 | 0}`

Specifies whether to analyze and report electromigration risk. This is a different type of analysis than the EM violation analysis performed by the Encounter power analysis software. Set the value to `1` to perform analysis.

Default: If you do not specify this parameter, this type of analysis is not performed.

`-analyzeFD {1 | 0}`

Analyzes and reports on filler density.

`-analyzeIR {1 | 0}`

Specifies whether to analyze and report IR drop. Set the value of this parameter to `0` to prevent IR drop analysis.

Default: If you do not specify this parameter, the software performs IR drop analysis.

`-analyzeIV {1 | 0}`

Specifies whether to analyze and report instance-based supply voltage. Set the value of this parameter to `1` to perform instance-based supply voltage analysis.

Default: If you do not specify this parameter, the software does not perform instance-based supply voltage analysis.

`-analyzePI {1 | 0}`

Specifies whether to analyze and report power-switch current. Power-switch analysis computes the ratio of current versus the saturation current (IDSAT) for each power switch in the design.

To enable power-switch current analysis, you must set the value of this parameter to `1` and specify the switched-on net using either the `-psoSONet` parameter of the `runVStorm` command or the VoltageStorm Analysis form (*Power – Analysis – Run VoltageStorm – Advanced* tab).

In this mode, the analysis treats all switches as on and estimates whether the power switches are capable of sustaining the current requirement of the switched block under specific operating conditions. A power current (PI) ratio greater than 1 means that there are power-switch violations which will require that you optimize the design to add or resize the power switches to fulfill the dynamic current requirements of the switched block.

`-analyzePV {1 | 0}`

Specifies whether to analyze and report power-switch voltage. Power-switch voltage analysis computes the IR drop across each power switch in the design. To enable power-switch voltage analysis, set the value of this parameter to `1`.

`-analyzeRC {1 | 0}`

Specifies whether to analyze and report resistor current. Set the value of this parameter to `1` to perform resistor current analysis.

Default: If you do not specify this parameter, the software does not perform resistor current analysis.

`-analyzeRJ {1 | 0}`

Specifies whether to analyze and report relative current density. Set the value of this parameter to `0` to prevent current density analysis.

Default: If you do not specify this parameter, the software performs relative current density analysis.

`-analyzeTC {1 | 0}`

Specifies whether to analyze and report tap current. Set the value of this parameter to `1` to perform tap current analysis.

Default: If you do not specify this parameter, the software does not perform tap current analysis.

`-analyzeVC {1 | 0}`

Specifies whether to analyze and report voltage source current. Set the value of this parameter to `1` to perform voltage source current analysis.

Default: If you do not specify this parameter, the software does not perform voltage source current analysis.

`-analyzeVU {1 | 0}`

Analyzes and reports on voltage drop across the package.

`-analyzeVV {1 | 0}`

Specifies whether to analyze and report voltages across vias and contacts. Set the value of this parameter to `0` to prevent analysis of voltages across vias.

Default: If you do not specify this parameter, the software performs analysis across vias and contacts.

`-biasVolt` *`value`*

Defines operating bias voltage with which rail analysis is performed.

Note: The bias voltage value is *always* the voltage value for the power net, whether you are analyzing a ground net or power net. For zero voltage (VSS, GND) nets make sure this is assigned correctly, since it will be defaulted to 1.2V.

Cadence strongly recommends that you use this parameter with MSV designs, as the power analysis software provides more accurate results with this parameter.

`-cellPowerFile` *`filename`*

Specifies the name of the VoltageStorm cell power file.

Note: If you specify this parameter, you must also specify the `-totalPower` parameter.

`-cmd` *`commandFile`*

Loads a VoltageStorm command file. When specified, Encounter automatically sends the command file information to VoltageStorm to perform power analysis.

`-cpfMode` *`mode`*

Specifies the CPF mode for analysis when the CPF file is loaded. You can also specify the CPF mode using the drop-down menu in the VoltageStorm Analysis form in the GUI. After specifying the mode, the software automatically populates the specified on and off nets for power switches.

`-currentFile {`*`fileNames`*`}`

Specifies the static or dynamic current files that were generated by the power analysis `report_power` command or the `-runPM` parameter (PowerMeter) of the `runVStorm` command. When specifying multiple files, include a space between each file name.

Note: You must specify the static or dynamic current files for always-on (`-psoAONet`) and switched-on nets (`-psoSONet`) for steady-state analysis (on) with power switches.

`-decapMethod {area | feasibility}`

Performs area-based or feasibility-aware decoupling capacitance optimization in dynamic mode.

Default: `area`

`area`:
Performs area-based decoupling capacitance optimization. After the successful completion of dynamic IR-Drop and decoupling capacitance analysis, VoltageStorm generates the ECO file (`vs2fe_eco.tcl`) for Encounter that is driven by the area-based `addDeCap` command. When you run the ECO file in Encounter using the `source vs2fe_eco.tcl` command, Encounter swaps the filler cells in the design with decoupling capacitance cells, then checks for DRC violations based on the technology LEF rules for the design. You can use the `reportDeCap` command to generate a report on the newly added decoupling capacitance cells in the design.

`feasibility`:
Performs feasibility-aware (also known as placement-aware) decoupling capacitance optimization. This method relies on the available filler cells in the design. During decoupling capacitance optimization in VoltageStorm, the additional capacitance value is computed for each node on the power grid. The analysis searches for available filler cells near the nodes that require additional decoupling capacitance and generates an ECO file that contains the filler instances that should be swapped with decoupling capacitance cells. The ECO file is then sourced again for reanalysis of the power grid. During re-analysis, VoltageStorm swaps the filler cells with decoupling capacitance cells and dynamic IR drop is analyzed to assess the impact of the newly added decoupling capacitance cells.

`-defaultFrequency` *`megahertz`*

Specifies the operating frequency of the design.

`-defVersion 5.4 | 5.5 | 5.6`

Specifies the DEF version to use.

`-dynamic` Specifies that VoltageStorm should run in dynamic mode.

`-emLifetime value`

Specifies the expected lifetime of the design in years. This information is used when assessing electromigration risk. The equivalent VoltageStorm variable is `lifetime`.

Default: `10`

`-emModelFile` *`fileName`*

Specifies the name of an EM Model file to load into the software.

Default: If you do not specify this parameter, EM Model files are not loaded.

`-genBbv {1 | 0}`

Specifies whether to generate a boundary voltage file (for hierarchical designs). Set the value to `1` to generate boundary voltages in a `.pp` file format for all macros in the design. Set the value to `0` to turn off boundary voltage generation. The equivalent VoltageStorm command is `write block_boundary_voltage`.

Default: `1`

`-genIV {1 | 0}`

Specifies whether to generate an instance voltage file. Set the value of `1` to generate a file named *`designName`*`.irdrop.custom.` Set the value to `1` to generate instance voltages. Set the value to `0` to turn off instance voltage file generation. The equivalent VoltageStorm command is `print iv` *`fileName`*

Default: `0`

`-genView {1 | 0}`

Specifies whether to generate a hierarchical power grid view. Set the value to `1` to generate a hierarchical power grid view. Set the value to `0` to turn off hierarchical power grid view generation. the value to `0` to turn off instance voltage file generation. The equivalent VoltageStorm command is `generate_view`.

Default: `1`

`-genViewLef` *`fileName`*

Specifies the name of the LEF file for the block for which to generate the hierarchical power grid view. For example, if you are analyzing a block named `Block1`, set the value of this parameter to the name of the LEF file that contains a macro statement for `Block1`.

`-genViewView {detailed | abstract | reduced}`

Selects the type of power grid view to generate. Specify one of the following.

`detailed`

Generates a detailed power grid view and a port power grid view of your design. These views are based on a completed VoltageStorm analysis and contain non-uniform tap current distributions.

`abstract`

Generates an abstract power grid view of your design based on the detailed view. These views are based on a completed VoltageStorm analysis and contain a detailed view and an abstract view. The abstract view is based on a mathematically equivalent impedance value assigned to the ports that represents a non-uniform tap current distribution.

`reduced`

Generates a static or dynamic reduced power-grid view from a detailed view using an RC reduction similar to that for abstract views. While the abstract view preserves only the impedance between the ports, the reduced view preserves the impedance between the ports, as well as that among all the tap nodes, that is, all the nodes that consume current.

The reduced view enables you to check for IR drop at all nodes that consume current.

Default: `detailed`

`-isPM {1 | 0}` Indicates whether the cell power file is in PowerMeter format.

`-ivFile` *`fileName`*

Generates an instance voltage file as output for static or dynamic IR-drop analysis. You can pass the instance voltage file to the Cadence® Encounter® Timing System (ETS) or CeltIC® NDC to perform IR-drop aware delay calculation and timing analysis.

Note: To enable this functionality, you must specify the *IV* analysis type for instance-based supply voltage analysis using the `-analyzeIV` parameter or the VoltageStorm Analysis form in the GUI.

Tip

When performing IR-drop aware timing analysis, you can improve timing accuracy by specifying the tri-voltage timing libraries instead of using the k-factor based timing libraries. The tri-voltage libraries are characterized with a minimum of three voltages for the same corner. The linear deration that occurs with k-factor based timing libraries may compromise the accuracy of the delay induced by the IR drop.

`-keepTempFiles {1 | 0}`

Specifies whether to retain temporary data files in the run directory. Set the value to `1` to retain these files. Set the value to `0` to delete these files by the end of a VoltageStorm run.

Note: Use this parameter to retain necessary input files for running VoltageStorm in standalone mode. These files are removed after the `runVStorm` command is executed.

Default: `0`

`-layerBias` *`lefLayer value`*

Sets a bias value, in microns, by which to undersize all wires on a defined layer. Specify a value in the range of -1 to 1. This value is applied to each edge of the wire, decreasing the width by two times the bias value. This value affects the resistance extraction of the power grid.

Default: `0`

`-layerMap` *`lefLayer icecapLayer`*

Maps the layer names in the IceCaps technology file to the layer names in the LEF technology file. You only need to use this command if the layer names specified in the IceCaps technology file are different from those used in the LEF file.

`-libs` *`dirNameList`*

Specifies a list of the cell libraries containing power grid views.

Note: You *must* specify libraries in `-dynamic` mode.

`-msmvBiasVolt {nets}`

Specifies the operating voltage for each multiple supply multiple voltage (MSMV) net in the design. When specifying the operating voltages, separate each operating voltage with a space.

Note: This parameter is required when running dynamic power analysis in conjunction with the `-runPM 1` parameter of the `runVStorm` command (or when the PowerMeter (*PM*) button is enabled in the VoltageStorm Analysis form of the GUI).

`-msmvNet {`*`nets`*`}` Specifies the MSMV net(s) in the design. When specifying multiple MSMV nets, include a space between each net.

Note: This parameter is required when running dynamic power analysis in conjunction with the `-runPM 1` parameter of the `runVStorm` command (or when the PowerMeter (*PM*) button is enabled in the VoltageStorm Analysis form of the GUI).

`-net` *`name`* Specifies the net name for which VoltageStorm analyzes power.

The first net in the list of power net names (as defined in the configuration file) is analyzed.

Note: You must specify nets in `-dynamic` mode.

`-netVolt` *`value`* The voltage of the net in volts.

`-operationMode` *`mode`*

Specifies the operation mode. When used with the `-vrangePercent` parameter, you can specify a voltage variation range that is used to characterize the libraries in sign-off mode.

`-outDir` *`dirName`*

Specifies the name of the directory in which to store the output directories resulting from analysis.

VoltageStorm places the results for a specific analysis in the state directory below the output directory.

The equivalent VoltageStorm variable is `output_directory_name`.

Default: If you do not specify an output directory, VoltageStorm places the results for each specific analysis in the run directory.

`-packMappingFile` *`fileName`*

Specifies the name of the package terminal mapping file, which contains the mapping of the terminals of the SPICE subcircuit in the package model file (specified by the `packModelFile` option) to the locations at the grid or top-level pins.

The format of the file is as follows:

```
subckt_terminal_name
subckt_terminal_name pin toplevel_pin_name
subckt_terminal_name cell pad_cell_instance_name
subckt_terminal_name x,y
subckt_terminal_name [power|ground]
```

`-packModelFile` *`fileName`*

Specifies the name of the package model file which contains a SPICE subcircuit describing a package model. It enables the use package models during full-chip static and dynamic power-grid analysis. This file is generated by Allegro Package SI (also known as APE3D by Cadence) or by other package-modeling products.

`-pmDesignIncludeFile` *`fileName`*

Specifies the design include file that contains the PowerMeter command to specify a stimulus pin that the piecewise-linear (PWL) source will be applied to. For example:

```
design rSpiceDeckPWLPin "design or instance:PIN , PWL
( 0 0 400ps 0 600ps 0.8)"
```

`-pmPowerIncludeFile` *`fileName`*

Specifies the power include file that contains the PowerMeter command to generate the `static_`*`netname`*`.ptiavg` current data files when performing static power-consumption calculation with PowerMeter. For example:

```
ChipPwr rWriteStaticUTIFiles 1
```

PowerMeter creates a current data file for the following nets:

- Nets defined as power or ground nets in the DEF file
- Nets with three or more connections when more than half of the connections are power or ground nets
- Nets with two connections when both are power or ground nets
- Nets that you define as power rails

Note: PowerMeter does not use the routing data, only the connectivity data. If the net is connected by abutment and you have connectivity described in the DEF file, PowerMeter will generate the current data file. If the connectivity of abutted cells is not in the DEF file, PowerMeter will not see that these cells are connected to a power net and will not generate the current data file.

`-pmPowerOutIncludeFile` *`fileName`*

Specifies the power output include file that contains the PowerMeter command to report on the cell types in the design. You can report on the macro, I/O, combinational, sequential, or blackbox cell types. For example:

```
report_power -outfile power.report -cell_type { macro |
io | combinational | sequential | blackbox }
```

`-powerFile` *`fileName`*

Specifies the name of the input file containing the instance-based power consumption estimates for all instances of each cell and block in your design. You can generate this file by issuing the `updatePower` command with the `-reportInstancePower` option from the Encounter console.

Note: If you do not specify this parameter, you must specify the `-totalPower` parameter.

`-powerNetGroup {`*`nets`*`}`

Enables you to group power nets when running power analysis on power-switched designs.

Note: To enable this functionality, you must load the CPF file to automatically populate the power net groups. If you do not load the CPF file, the always-on (`-psoAONet`), switched-on (`-psoSONet`), or switched-off (`-psoSFNet`) nets control the grouping of power nets.

`-powerUp {1 | 0}`

Specifies whether to enable the power-up analysis of a switched net in the power domain. Set this parameter to `1` to enable power-up analysis.

Note: To run power-up analysis, you must also specify the switched-on net using the `-psoSONet` parameter and enable PowerMeter using the `-runPM 1` parameter. You can also use the VoltageStorm Analysis form (*Power – Analysis – Run VoltageStorm – Advanced* tab) to perform these operations.

`-powerUpSpiceCorners` *`corners`*

Specifies the device corners for power-up analysis. When specifying the device corners, include a space between each corner. The device corners are required to run power-up analysis.

`-powerUpSpiceModels` *`fileNames`*

Specifies the device SPICE models for power-up analysis. When specifying the device SPICE models, include a space between each SPICE model. The device SPICE models are required to run power-up analysis.

`-powerUpSpiceNetlists` *`fileNames`*

Specifies the SPICE netlists for power-up analysis. The SPICE netlists are optional if you have already included a detailed-view library (cell library generated by GDS) during analysis. If the LEF-only library was used for analysis, you must provide an external SPICE subcircuit for each cell in the design for power-up analysis.

`-powerUpSpicePWL " time1 voltage1 time2 voltage2..."`

Specifies the piecewise-linear (PWL) source that is applied to the traced control pin of the design to power-up the switched net. For example:

```
-powerUpSpicePWL "0 0 400ps 0 600ps 0.8"
```

You can view the traced stimulus file (*switchedNet*`.stimulus`) after dynamic power calculation completes. You can also supply a stimulus pin using the `-pmDesignIncludeFile` parameter and command. For example:

```
design rSpiceDeckPWLPin "design or instance:PIN , PWL
( 0 0 400ps 0 600ps 0.8)"
```

`-ppFile` *fileName*

Specifies the name of the power pad location file. This file can be in Encounter format or VoltageStorm format. If the `runVStorm` command detects a power pad location file in Encounter format, `runVStorm` automatically converts the file to VoltageStorm format.

The minimum information that this file must specify is the x and y coordinates and the layer name.

`-psoAOIR` *limit*
Specifies the IR-drop limit for the always-on net (`-psoAONet`).

`-psoAONet` *net*
Specifies the always-on net during the analysis of a power-switched design. If the CPF file is loaded, the software automatically populates the *Always-on Net* setting in the VoltageStorm Analysis form.

`-psoAOVolt` *voltage*

Specifies the always-on net voltage. If the CPF file is loaded, the software automatically populates the *Always-on Voltage* setting in the VoltageStorm Analysis form.

`-psoSFNet` *net*
Specifies the switched-off net during the analysis of a power-switched design. If the CPF file is loaded, the software automatically populates the *Switched (OFF) Net* setting in the VoltageStorm Analysis form.

`-psoSONet` *net*
Specifies the switched-on net during the analysis of a power-switched design. If the CPF file is loaded, the software automatically populates the *Switched (ON) Net* setting in the VoltageStorm Analysis form.

`-runECO {1 | 0}` Instructs VoltageStorm to run ECO functionality after power analysis.

`-runPM {1 | 0}` Specifies that the power analysis software should start a PowerMeter run.

`-simResolution` *`value`*

Specifies the simulation time step to be used.

`-simPeriod` *`value`*

Specifies the total simulation time to be used during power computation.

`-spef` *`fileName`* Specifies the name of a parasitics file in DSPF or SPEF format.

By default, VoltageStorm automatically extracts parasitics. If you use this parameter, VoltageStorm uses the parasitics data from the file rather than performing RC extraction to generate the parasitics.

`-static` Specifies that VoltageStorm should run in static mode. This is the default mode.

`-tempDir` *`dirName`*

Specifies the name of the directory to which temporary files are written. If you do not specify a location, temporary or unimportant intermediate files are written to the `/tmp` directory. If you do not have write permission to the `/tmp` directory, you must specify another location in order to run VoltageStorm.

`-temperature` *`degreesCelsius`*

Specifies a temperature value in degrees Celsius.

Default: `25`

`-totalPower` *`watts`*

Specifies a voltage value to be used in conjunction with the VoltageStorm cell power file. This parameter is required if you specify the `-cellPowerFile` parameter.

Note: If you do not specify this parameter, you must specify the `-powerFile` parameter.

`-twf` *`fileName`*

Specifies the timing window file.

By default, VoltageStorm automatically generates a timing window file. If you specify this parameter, VoltageStorm uses the specified file rather than generating a timing window file.

`-useCellView {`*`type name view`*`}+`

Selects the power grid view to use in modeling the cell characteristics. In the VoltageStorm log file, VoltageStorm records a list of cells and the view type used for the analysis.

You must specify a type, name, and view.

For *`type`*, specify one of the following:

- `file`
- `pattern`

For *`name`*, specify the name of the file.

For *`view`*, specify one of the following:

- `accurate`
- `abstract`
- `fast`
- `floorplan`
- `port`
- `reduced`
- `detailed`

Default: If you do not specify a view, VoltageStorm uses the `accurate` view, that is, the most accurate view for each cell available in the library.

The following examples show how to specify the *type*, *file*, and *view*.

`file` *`cell_list`*`.file port`

Specifies that all cells contained in the file *`cell_list`*`.file` should use the port view.

`pattern *`*`AD`*`* port`

Specifies that any cell matching the pattern *`AD`* would use the `port` view. For example, a cell named `ADDFHX1` matches this pattern and uses the `port` view. You can use `pattern` to specify a particular cell that uses a particular view.

`pattern` *`cellname`* `port`

Specifies that only the named cell would use the `port` view.

`pattern ADDFHX1 port`

Specifies that the cell named `ADDFHX1` uses the `port` view.

If you want to run VoltageStorm with a different set of views for individual cells, you must use this parameter to specify those cells.

`-vcdBased {1 | 0}`

Controls whether vectorless (`0`) or vector-based (`1`) dynamic power calculation is performed.

Default: `0`

`-vcdFile` *`fileName`*

Specifies the name of the VCD file for use in vector-based dynamic power calculation.

`-vcdScope` *`scopeName`*

Specifies the design's hierarchical scope to be used in the VCD file.

`-vcdStartTime` *`nanoseconds`*

Specifies the starting time to be used in the VCD file.

`-vcdStopTime` *`nanoseconds`*

Specifies the stop time to be used in the VCD file.

`-view` *`value`*

Specifies the view that was created using the multi-mode multi-corner (MMMC) `create_analysis_view` command. Power analysis is performed for the specified view.

`-voltLimit` *`voltage`*

Specifies the voltage threshold on the power or ground net.

Default: If you do not specify this parameter, VoltageStorm uses a value that is 10% less than the nominal power voltage.

`-vrangePercent` *`value`*

Specifies the voltage variation range. When used with the `-operationMode` parameter, you can specify a voltage variation range that is used to characterize the libraries in sign-off mode.

vsBeginIncludeFile *`filename`*

Specifies the name of the the file that includes power analysis commands that will be executed during the run.

`-workDir` *`dirName`*

Specifies the name of the directory into which VoltageStorm places the output data files. Specify one of the following:

- The RC database generated from the DEF resistance extraction by QX
- The automatically generated extraction command file
- A file containing a list of the cells and views used for the run that generated the existing data
- A file that lists the libraries specified and linked by the `library_list` command
- A single `lib_merge.cl` library resulting from the linking of the libraries specified by the `library_list` command
- A *`netname.`*`uncon` file if the DEF file contains a logical connection to an instance, but no physical connection to the instance exists in the design

The equivalent VoltageStorm variable is `work_directory_name`.

Default: If you do not specify this parameter, a directory named `work` is created in the run directory, and the files are written to that `work` directory.

Example

- The following command runs VoltageStorm IR drop, relative current density, and via and contact voltage analysis for net VDD, using the following files:
  - A power grid view library named `pglib_18`
  - An instance power file named `instance.power.VDD`
  - A power pad/pin location file named `dtmf_chip_VDD.pp`

```
runVstorm -net VDD -libs ./pglib_18.cl -powerFile instance.power.VDD
-ppFile dtmf_chip_VDD.pp
```

saveEM

```
saveEM
     [-outDir dirleName]
     -net netName
     [-ascii]
```

Writes an EM file for the power net that was analyzed and displayed. The file contains the EM display information and is used to redisplay the EM analysis. Use this command after performing early static rail analysis using the `updatePower` command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-ascii`	Saves the EM file in text format. This option can be useful during troubleshooting. **Note:** If you use this option, you cannot reload the saved ASCII file.
`-net` *`netName`*	Specifies the net name for which the EM file contains information.
`-outDir` *`dirName`*	Specifies the name of the directory to which the IR drop information is saved. This file is in binary format.

Examples

- The following command writes out the displayed EM information for VDD to a file named `VDD.em` in the `emoutput` directory:

  ```
  saveEM -outDir emoutput -net VDD
  ```

- The following command writes out the displayed EM information for VDD, in ASCII format, to a file named `VDD.em.ascii`.

  ```
  saveEM -outDir emoutput -net VDD -ascii
  ```

  The format of the file is shown in the following example:

  ```
  LayerName: M1  MaxEM: 0.831666  MinEM: 0  Threshold: 0.831666
  LayerName: M5  MaxEM: 0.164652  MinEM: 0  Threshold: 0.16
  LayerName: M6  MaxEM: 0.595251  MinEM: 0  Threshold: 0.59
  ```

```
LayerName: V12 MaxEM: 0.107329  MinEM: 0  Threshold: 0.1
LayerName: V23 MaxEM: 0.107329  MinEM: 0  Threshold: 0.1
LayerName: V34  MaxEM: 0.107329  MinEM: 0  Threshold: 0.1
LayerName: V45  MaxEM: 0.135117  MinEM: 0  Threshold: 0.13
LayerName: V56  MaxEM: 0.180156  MinEM: 0  Threshold: 0.18
( 50020 48960 53020 51960 ) id:162 "V56" 0.054884 mA/cut
( 50020 42240 53020 48960 ) id:161 "M6" 0.164652 mA/u
( -52380 42240 -49380 48960 ) id:160 "M6" 0.21313 mA/u
( 50020 41440 53020 42240 ) id:159 "V56" 0.178882 mA/cut
...
```

Note: The coordinates enclosed within parentheses, such as `(50020 48960 53020 51960)`, are in database units.

saveIRDrop

```
saveIRDrop
     -outDir dirName
     [-net netName]
```

Writes an IR drop file for the power net that was analyzed and being displayed. The file contains the IR display information and is used to redisplay the IR drop information. Use this command after performing early static rail analysis using the `updatePower` command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-net` *`netName`*	Specifies the net name for which the IR drop file contains information. *Default:* If you do not specify this parameter, the software saves the IR drop information that is currently displayed.
`-outDir` *`dirName`*	Specifies the name of the directory to which the IR drop information is saved. This file is in binary format.

Example

- The following command writes out the displayed IR drop information for VDD to a file named `vdd1.ir` in a directory named `iroutput`.

  ```
  saveIRDrop -outDir iroutput
  ```

saveIVD

```
saveIVD
     -outDir dirName
     [-net netName]
```

Writes an Instance Voltage Drop file for the power net that was analyzed and being displayed. The file contains the instance voltage drop display information and is used to redisplay that information. Use this command after performing early static rail analysis using the `updatePower` command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-net` *`netName`*	Specifies the net name for which the Instance Voltage Drop file contains information. *Default:* If you do not specify this parameter, the software saves the instance voltage drop information that is currently displayed.
`-outDir` *`dirName`*	Specifies the name of the directory to which the instance voltage drop information is saved. This file is in binary format.

Example

- The following command writes out the displayed instance voltage drop information for VDD to a file named `VDD.ivd` in a directory named `ivdoutput`.

  ```
  saveIVD -outDir ivdoutput -net VDD
  ```

savePadLocation

```
savePadLocation
     -outfile fileName
     [-lef]
     [-keepPadLocDisplay]
     [-VSformat]
     {x y [layerName [padInstName | voltage]] }
```

Saves the power pad location data, in ASCII format, to run rail analysis. The power pads are created by the Edit Pad Location form (*Power - Analysis - Edit Pad Location*). Use this command after adding or automatically fetching pad locations.

Parameters

`-keepPadLocDisplay`	Keeps the power pad location data in memory and in the GUI. Use this parameter after using the Edit Pad Location and Display Pad Location forms. The power analysis software automatically saves the power pad location data that is displayed in the GUI to a reserved file named `paAutoGen.`*`netName`*`.pp` and uses this file to perform rail analysis.
`layerName`	Specifies the generic metal routing layer name. Use a value such as `M1` or `V12`.
`-lef`	Specifies the layer name defined in the LEF file instead of the generic metal routing layer name, such as `M1` or `V12`.
`-outfile` *`fileName`*	Specifies the name of the file to which pad location data is written. The file is in ASCII format.
`padInstName`	Specifies the name of the power or ground pad instance at the design level.
`voltage`	Specifies the boundary voltage at the block level.
`-VSformat`	saves the pad location file in the format that VS (EPS) accepts.
`x y`	Specifies the x and y coordinates of the power pad location. Units in microns.

Example

- The following command saves the `vdd` pad location x and y coordinates and layer data to a file named `vdd.pp`:

  ```
  savePadLocation -outfile vdd.pp
  ```

  Note: Optionally, if pad instance names were in memory, or if boundary voltages were known, they would also be saved into the file. For example, if you issue the `autoFetchDCSources` command at the chip level, where I/O power pads are defined, the pad instance names would be in memory. Or if power analysis is run in a hierarchical design and partitions are saved, boundary voltages would be in memory.

saveToggleProbability

```
saveToggleProbability
     -outfile fileName  {clockName clockRate toggleProb}
```

Saves the toggle probability information, in ASCII format. Use this command after extracting the RC data in your design.

The format of the Net Toggle Probability file is:

```
clockName clockRate toggleProbability
```

The following example shows the contents of a Net Toggle Probability file that has three clocks:

```
vclk1 25.000 0.200
vclk2 52.500 0.200
vclk3 85.500 0.200
```

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

clockName	Specifies the name of the clock from the timing constraints file that was read in during design import.
clockRate	Specifies the toggle rate of the specified clock.
-outfile *fileName*	Specifies the name of the file to which toggle probability data is written. The file is in ASCII format.
toggleProb	Specifies the toggle probability of nets as a real number. For nets, a toggle implies a switch from 0 to 1 or from 1 to 0. A value of 0.1, for example, indicates that 10% of the nets toggle at a given time. *Default:* `0.2`

Example

- The following command saves the toggle probability information for click `vclk1` to `togglefile.tg`:

  ```
  saveToggleProbability -outfile togglefile.tg {vclk1 25.000 0.200}
  ```

setEnergyLUTHandling

```
setEnergyLUTHandling {resetNegativeEnergy | keepNegativeEnergy}
```

Specifies whether or not the power analysis software honors negative energy from the power library look up table (LUT) when calculating power consumption. By default, the power analysis software honors negative energy in the power library. Use this command after extracting the RC data in your design and before you issue the `updatePower` command.

Note: If the extrapolation from positive values yields a negative result, that negative value is always reset to `0`.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`keepNegativeEnergy`

Honors negative energy values from the power library look up table.

`resetNegativeEnergy`

Always resets negative values to zero.

setLibraryPowerUnit

`setLibraryPowerUnit -libname` *`libraryName`* `-unit` *`unitValue`*

Sets the timing library power unit value after the timing library file has been read. If the timing library file already defines a power unit value, the power unit value cannot be changed. To report the power unit value, use the getLibraryPowerUnit command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-libname` *`libraryName`*

Specifies the name of the timing library file.

`-unit` *`unitValue`*

Specifies the timing library power unit value. You can specify one of the following standard units: 1mW, 100uW, 10uW, 1uW, 100nW, 10nW, 1nW, 100pW, 10pW, or 1pW.

Example

- The following command sets the power unit value of the `tim.lib` library to `10nW`:

```
setLibraryPowerUnit -libname tim.lib -unit 10nW
```

setPowerAnalysisLibrary

```
setPowerAnalysisLibrary
     {-view viewName | {-internalPowerView viewName1 -leakagePowerView
     viewName2}}
     |
     { {min | max} | {-internalPower {min | max} -leakagePower {min | max} }
```

Specifies the timing library groups to use for internal and leakage power by the power analysis software. Use this command after extracting the RC data in your design and before issuing the `updatePower` command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-internalPower {min | max}`

Uses the timing library groups for internal power.

`-internalPowerView` *viewName1*

Uses the timing library groups from the specified analysis view for internal power. Use this parameter when the software is in multi-mode multi-corner analysis mode.

`-leakagePower {min | max}`

Uses the timing library groups for leakage power.

`-leakagePowerView` *viewName2*

Uses the timing library groups from the specified analysis view for leakage power. Use this parameter when the software is in multi-mode multi-corner analysis mode.

`min | max` Uses the minimum or maximum timing library group for both internal and leakage power.

`-view` *ViewName* Uses the timing library groups from the specified analysis view for both internal and leakage power. Use this parameter when the software is in multi-mode multi-corner analysis mode.

setPowerAnalysisSlew

```
setPowerAnalysisSlew {best | worst}
```

Specifies whether to set a slew value based on an ideal clock or to set a slew value that simulates a clock with a slower slew. Use this command after extracting the RC data in your design and before issuing the `updatePower` command.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`best`	Sets slew values based on an ideal clock, even if clock trees are present in the design. Since a faster slew draws the largest current, using this set of slew values leads to the highest (most pessimistic) IR drop value.
`worst`	Sets slew values that simulates a clock with a slower slew than an ideal clock. Since a slower slew draws less current, using this set of slew values leads to a less pessimistic IR drop value.

updatePower

```
updatePower
     {-vcd vcdFileName [-vcdTop VCDTopName] [-start time] [-end time]
               [-timeUnit unit] [-noTop]
          | -preCTS
               -toggleFile toggleFileName
               [-expectedPower power]
          | -postCTS
               -toggleFile toggleFileName
               [-expectedPower power]
          | -toggleProb probability
               -clockRate clockrate
               [-expectedPower power]
          | -estimate filename}
          | -totalPower value
     [-readInstancePower fileName]
     {-noRailAnalysis | -pad padFileList}
     [-temperature degreesCelsius]
     [-mode {floorplan | layout}]
     [-irDropAnalysis {average | peak}]
     [-report fileName]
     [-reportInstanceVoltage fileName]
     [-reportInstancePower fileName]
     [-reportRailAnalysis fileName]
     [-reportNetPower fileName]
     [-biasVoltage volts]
     [-isSignal]
     [-reportDCSourceCurrent fileName]
     [-reportUnclockedInstances fileName]
     [-readBlockPinCurrent fileName]
     [-readCellPower cellPowerFileName]
     [-readPackageModel packageModelFile]
     [-readPackageTerminalMapping packageTerminalMappingFile]
     [-readPadResistance padResistanceFile]
     [-readPwrGatingCell powerGatingFile]
     [-group netsList]
     [-tcf tcfFileName]
     [-preCTS | -postCTS | -clockRate clockrate | -toggleProb prob]
     powerNetNames
     [-extend]
```

Starts the power analysis on the power nets. If you are using a VCD file to run a simulation-based power analysis, use this command after issuing the probePower command. If you are running a statistical power analysis, use this command after extracting the RC data.

The power analysis software uses the common library group used during the timing analysis stage of the flow.

Note: If you do not want to toggle data paths, but want to toggle clock trees at 100%, use a small toggle rate for the data paths (for example, 0.0001, or 0.01%). This data path toggle setting makes the power analysis tool toggle a very small percentage of data nets, but the resulting amount of power used is not very different than the power consumed by the clock tree. Using even smaller toggle rates will increase accuracy.

Note: This command is obsolete and will continue to be supported in this release, but will be removed in the next major release of the software. Use the new Power-Grid Library Commands and Rail Analysis Commands.

Parameters

`-biasVoltage` *`volts`*
: Specifies the a voltage value to be applied to power and rail calculations.

  Note: This value affects only the switching power.

`-clockRate` *`clockrate`*
: Specifies the clocking frequency, in megahertz. This parameter is used in conjunction with the `-toggleProb` parameter for a statistical power analysis.

`-end` *`time`*
: Specifies the end time for power analysis when you are using a VCD file. This parameter is used in conjunction with the `-vcd` and `-start` parameters for a simulation-based power analysis.

  Default: If you do not specify this parameter, the start and end time is the entire simulation time in the VCD file.

`-estimate` *`filename`*

Specifies that power analysis is to be performed in a simple estimate mode. The *`filename`* specifies a file that contains the following keywords along with a value for each keyword:

- `clock_rate`
- `toggle_probability`
- `load_capacitance`
- `multiplier`
- `slew`

Issue the following command to perform rail analysis with the power number obtained using the simple estimate mode power calculation:

```
updatePower -pad padFileName -estimate filename powerNetName
```

Issue the following command to perform a simple estimate mode power calculation without rail analysis:

```
updatePower -noRailAnalysis -estimate filename powerNetName
```

Issue the following command to read in an instance power file. This overwrites the power number that is generated when you perform a simple estimate mode power calculation for each instance to calculate the total consumed power:

```
updatePower -readInstancePower filename -estimate filename powerNetName
```

Issue the following command to write the power number (obtained by the simple estimate mode power calculation) to an instance power file:

```
updatePower -reportInstancePower filename -estimate filename powerNetName
```

`-expectedPower` *`power`*

Specifies the expected power, in milliwatts, and directs the program to estimate a toggle percentage value. This parameter is used for a statistical power analysis.

`-extend`

Creates virtual followpins on rows not containing standard cells (floorplan mode).

Note: By default, the power analysis software creates virtual followpins for all rows containing standard cells.

`-group` *`netsList`*

Specifies a group of power nets that are connected through power gating cells. The always-on net must be included in the net list.

`-irDropAnalysis {average | peak}`

Specifies whether to perform IR drop analysis in average mode or peak mode. Specify one of the following:

`average`

Displays a gradient of the average IR drop during the entire analysis period. This is the default. It helps identify power grid integrity problems.

`peak`

Displays a gradient that shows the largest IR drop during the analysis period. This helps identify IR drop that is caused by local switching activity.

`-isSignal`

Instructs the power analysis software to analyze signal nets for DC electromigration. This parameter is useful *only* for signal nets that behave like power/ground nets—such as nets that connect analog blocks.

Use the pad location file to define the driver terminals of such signal nets.

Use the instance power file to specify the power consumption of the receiver terminals of such signal nets.

Note: This parameter does *not* instruct the software to analyze power or ground nets.

Important

When you use the `-isSignal` parameter, you must also use these `updatePower` parameters:

- `-biasVoltage`
- `-mode layout`

`-mode {floorplan | layout}`

Specifies the mode of running power analysis. Specify one of the following:

`floorplan`

Analyzes power in the early stage of floorplanning using power and ground mesh.

`layout`

Note: Layout mode is obsolete. It will continue to be supported in this release, but will be removed in the next major release of the software. To sign-off on the power grid, use VoltageStorm instead.

Analyzes power when the power and ground routes are actually designed.

Default: If you do not specify this parameter, power analysis runs in floorplan mode.

`-noRailAnalysis`

Prevents the power analysis software from running rail analysis. Use this parameter to perform power analysis based on logical connectivity. If you select this option, the following parameters are not used:

- `-mode`
- `-irDropAnalysis`
- `-reportInstanceVoltage`

This parameter is mutually exclusive with the `-pad` parameter.

`-noTop`

Turns off probing of the top-level module. This parameter is used for a simulation-based power analysis.

Default: If you do not specify this parameter, the top-level is included in the probing.

`-pad` *`padFileNames`*

Specifies the names of the power pad files. Separate multiple file names with a space. These files contain the specified power pad locations of the power source. Use this parameter to perform power and rail analysis based on physical connectivity. This parameter is mutually exclusive with the `-noRailAnalysis` parameter.

Note: You must specify the files for the `-pad` parameter in the same sequence as the corresponding *`powerNetNames`*. If the pad files are for power or ground pads, they cannot be shared among different nets.

`-postCTS`

Performs power analysis with clock nets at a set clock rate (in megahertz) for each clock domain, and a set toggle probability (in percent) for the clock's data paths. This means that all the nets on a clock path toggle at the clock rate, but only a percentage of the clock's data paths toggle in each clock cycle.

For example, if the *Clock Rate* value is `300` and the *net toggle probability* value is `0.2` (20%), then the clock path nets toggle at 300 MHz, and only 20 percent of the clock's data paths toggle in each cycle. Nets not belonging to a clock domain do not toggle.

If you do not specify this parameter or the `-preCTS` parameter, a dummy clock is used.

`powerNetNames`

Specifies the names of the power or ground nets to be analyzed. Separate multiple net names with a space.

You *must* specify the powerNetNames parameter as the last entry in an `updatePower` command string.

Note: You must specify the *`powerNetNames`* in the same sequence as the corresponding files that you specify for the `-pad` parameter.

`-preCTS`

Performs power analysis that is similar to that performed if you specify the `-postCTS` parameter, except that the power analysis software does not calculate switching power for nets with greater than 50 terminals. This behavior prevents the reporting of false IR drop in rail analysis.

If you do not specify this parameter or the `-postCTS` parameter, a dummy clock is used.

`-readBlockPinCurrent` *`filename`*

Specifies the name of a block power model file (`.ppi`).

During hierarchical power analysis, you obtain higher accuracy with *both* a block power model file *and* a macro power file.

Import a macro power file using the `-readInstancePower` parameter.

`-readCellPower` *`cellPowerFileName`*

Assigns power values to all instances of a cell, based upon a cell power file.

The cell power file has this format:

`cellName power_in_watts`

The following example cell power file entry specifies that all instances of cell `and2x1` should have the specified power value:

```
and2x1 1.0E-12
```

`-readInstancePower` *`filename`*

Specifies a text file that contains instance names and power consumption, in watts. You can use this parameter to specify predetermined power values for IP blocks, or for any instance in the design in which a user-specified value overrides the calculated value. The software also includes the blackboxes and blackblobs that are specified in the instance power file for rail analysis.

`-readPackageModel` *`packageModelFile`*

Reads the specified RLC package model file that contains the two-ports SPICE subcircuit describing the package model. Use this parameter to improve the accuracy of IR-drop analysis. You can create the RLC package model file using the Allegro® PCB SI XL tool.

Note: You must use this parameter in conjunction with the `-readPackageTerminalMapping` parameter.

`-readPackageTerminalMapping` *`packageTerminalMappingFile`*

Reads the specified RLC package terminal mapping file that contains the mappings for all terminals in the SPICE subcircuit of the RLC package model file. Use this parameter to improve the accuracy of IR-drop analysis.

Note: You must use this parameter in conjunction with the `-readPackageModel` parameter.

You can create the RLC package terminal mapping file manually or by using the Allegro® PCB SI XL tool. To create this file manually, use the following syntax:

`subckt_terminal_name` `cell` *`pad_cell_name`*

where:

- *`subckt_terminal_name`* specifies the name of the terminal that is being mapped in the SPICE subcircuit of the RLC package model file.
- `cell` *`pad_cell_name`* specifies the name of the cell that the terminal connects to in the SPICE subcircuit of the RLC package model file.

`-readPadResistance` *`padResistanceFile`*

Reads the specified pad resistance file that contains the pad resistance of all pads in the SPICE subcircuit of the RLC package model file. Use this parameter to improve the accuracy of IR-drop analysis.

Note: Before using this parameter, you must manually create the pad resistance file using the following syntax:

`pad_name` `R` *`resistance_value`*

where:

- *`pad_name`* specifies the name of the pad in the SPICE subcircuit of the RLC package model file.
- *`resistance_value`* specifies the resistance value (in ohms) of the pad in the SPICE subcircuit of the RLC package model file.

`-readPwrGatingCell` *`powerGatingFile`*

Reads the specified power-gating cell file to support multiple VDD nets that are connected by these power-gating cells.

The power-gating file uses the following syntax:

```
CELL cellname SUPPLY unswitched_pin_name
SWITCHED switched_pin_name RON value1_in_ohms
IDSAT value2_in_mA ILEAK value_in_mA
```

For example:

```
CELL HEAD16DM SUPPLY VDDG SWITCHED VDD RON 140 IDSAT 1.0
ILEAK 1e-3
```

Note: This parameter must be used in conjunction with the `-group` parameter to support the rail analysis flow.

`-report` *`filename`*

Specifies the name of the report file in which power analysis output information is written.

Default: If you do not specify this parameter, the report is written to the file *`powerNetName`*`.report`.

`-reportDCSourceCurrent` *`filename`*

Generates a block power model file for the block under analysis.

A block power model captures the relative distribution of current at the DC sources. The pad file contains the x and y coordinates, layer names, and (optionally) the pad instance names or boundary voltage values of the vdd pins. on page 902 illustrates the syntax of a block power model file.

Default: The default filename is *`block_name.net_name.`*`ppi`.

`-reportInstancePower` *`filename`*

Directs the power analysis software to generate a report that contains the instance name followed by the power consumption, in watts.

`-reportInstanceVoltage` *`fileName`*

Directs the power analysis software to generate an ASCII file containing instance voltages that can be used by any delay calculator to perform instance voltage-based delay calculations.

`-reportNetPower` *`filename`*

Directs the power analysis software to generate a report that shows the load capacitance, toggle rate, and switching power values for each net specified in the file.

Note: This parameter is only valid with statistical or simulation mode power analysis. You cannot use this parameter if you use the `-estimate` parameter.

`-reportRailAnalysis` *`filename`*

Directs the power analysis software to generate a report that contains IR drop and EM violation results on vias, wires, and instances.

`-reportUnclockedInstances` *`fileName`*

Reports the list of unclocked instances in the design (if applicable). The format of this file is:

`_unclocked_: instanceName`

Note: This parameter must be used in combination with the `-toggleFile` parameter.

`-start time`

Specifies the start time for power analysis when you are using a VCD file. This parameter is used in conjunction with the `-vcd` and `-end` parameters for a simulation-based power analysis.

Default: If you do not specify this parameter, the start and end time is the entire simulation time in the VCD file.

`-tcf tcfFileName`

Specifies the name of a TCF file.

For more information on TCF file syntax, see *Low Power in Encounter™ RTL Compiler*.

`-temperature degreesCelsius`

Specifies the temperature to be used during temperature-dependent resistance extraction. The temperature-dependent resistance coefficient is taken from the capacitance table.

`-timeUnit {ps | ns}`

Specifies the time unit of the VCD file. This parameter is used in conjunction with the `-start` and `-stop` parameters for a simulation-based power analysis.

Default: If you do not specify this parameter the time unit is picoseconds.

`-toggleFile toggleFileName`

Provides the name of the file that specifies each clock domain's clock frequency and toggle probability. The file can be generated using the Edit Net Toggle Probability form, and is used during analysis with real pre-CTS clock or real post-CTS clock data. This parameter is used for a statistical power analysis.

This parameter is only used if you specify either `-preCTS` or `-postCTs`. If you do not specify this parameter, a dummy clock is used.

`-togglеProb` *`probability`*

Specifies the toggle probability of nets as a real number. For nets, a toggle implies a switch from 0 to 1 or from 1 to 0. This parameter is used in conjunction with the `-clockRate` parameter for a statistical power analysis. A value of `0.1`, for example, indicates that 10% of the nets toggle at a given time.

Default: `0.2`

`-totalPower` *`value`*
: Specifies the power value, in milliwatts, used for running an area-based rail analysis. You do not need an instance power file to run in this mode. The software also includes blackboxes and blackblobs in the rail analysis.

`-vcd` *`vcdFileName`*
: Specifies the name of the VCD file. This parameter is used for a simulation-based power analysis.

`-vcdTop` *`VCDTopName`*

Specifies the top level of a VCD file. This parameter is only used in conjunction with the `-vcd` parameter for a simulation-based power analysis.

Examples

- The following command runs a simulation-based power analysis for `vdd` using the specified VCD and pad files:

  ```
  probePower
  updatePower -vcd /net/spc/design.vcd -pad /net/spc/pad.pp vdd
  ```

  The pad file contains the x and y coordinates, layer names, and (optionally) the pad instance names or boundary voltage values of the vdd pins.

- The following command runs a statistical power analysis for vdd using a toggle probability of 20% and a clock rate of 100 MHz:

  ```
  updatePower -toggleProb 0.2 -clockRate 100 -pad /net/spc/pad.pp vdd
  ```

  The pad file contains the x and y coordinates, layer names, and (optionally) the pad instance names or boundary voltage values of the vdd pins.

- The following command illustrates the block power model in a file `design.VSS.ppi`:

```
MACRO {
        NAME    design
        CURRENT 0.000310402
        CURRENT PIN     VSS     3.104024e-04
                0.115   10.08   M1      5.522246e-08
                0.115   110.88  M1      1.651548e-07
                0.14    151.2   M5      9.414273e-06
                0.115   20.16   M1      4.208810e-08
}
```

22

Rail Analysis Commands

analyze_early_rail

```
analyze_early_rail
     [-help]
     [-method static]
     -bias_voltage voltage
     -net_voltage voltage
     -volt_limit voltage
     [-temperature temperature]
     [-macro_power_file fileName]
     [-current_region_file fileName]
     -pad_location_file fileName
     [-power_gate_file fileName]
     [-power_grid_library directoryName]
     [-scale_hierarchy_current { {hier_name [scale|current] value} +}]
     [-display_IR]
     -net {netName +}
     [-instance_current_file fileName | -calculate_power | -total_current value]
     [-skip_virtual_via_on_layers [{{layer1 layer2} {layer3 layer4} ...} | all]]
```

This command is used to do a fast first rail analysis to find power grid integrity problems and to find potential hot spots. This command is used during the early stage of power grid design and is done during the floorplanning stage of layout.

Note: Default currents are in mA, power is in mW, and voltages are in volts.

Parameters

`-bias_voltage` *`voltage`*

Specifies the voltage between power and ground.

`-calculate_power` Specifies that the Common Power Engine (CPE) will calculate the power.

`-current_region_file` *`fileName`*

Specifies a file that includes a list of regions and the amount of current to be distributed within them for the power-grid. Default units in file are mA.

`-display_IR` Specifies that IR drip will be displayed.

`-help` Outputs a brief description that includes the type and default information for each `analyze_early_rail` parameter.

For a detailed description of the command and all of its parameters, use the man command `man analyze_early_rail`.

`-instance_current_file` *`fileName`*

Specifies the name of the instance current file generated by the Common Power Engine (`.ptiavg` file). This file shows the amount of current for each instance specified. Default units in this file are in mA.

`-macro_power_file` *`fileName`*

Specifies the name of the macro power file, when you want to specify pre-characterized power for custom macros.

`-method static`

Specifies the analysis type.

Default: `static`.

Currently only static is supported.

`-net {`*`netName`* `+}` Specifies a list of power or ground nets. Multiple nets can be specified, in order to analyze nets of the same voltage as a group.

`-net_voltage` *`voltage`*

Specifies the net voltage in volts.

`-pad_location_file` *`fileName`*

Specifies the name of a file that includes the pad locations.

`-power_gate_file` *`fileName`*

Specifies the name of the power gate (power switch) file that will be used to perform power gate steady-state analysis.

`-power_grid_library` *`directoryName`* `+`

Specifies the names of the power-grid view library directories.

`-scale_hierarchy_current {`*`hier_name`* `[scale | current] value +}`

Specifies a current or scale factor for the design hierarchy.

`-skip_virtual_via_on_layers [{{`*`layer1 layer2`*`} {`*`layer3 layer4`*`} ...} | all]`

Skips via insertion between stripes and non-stripes, on the specified LEF layer pairs.

`{`*`layer1 layer2`*`} {`*`layer3 layer4`*`} ...`

Specifies the LEF layer names.

`all`

Skips virtual via generation totally.

For example, `-skip_virtual_via_on_layers {{M1 M3} {M6 RDL}}` skips via stacking from `M1` to `M3` and `M6` to `RDL`.

`-temperature` *`temperature`*

Specifies the operating temperature in Celsius.

Default: 25^{o} C

`-total_current` *`value`*

Specifies the total current of the design in mA.

`-volt_limit` *`voltage`*

Specifies the minimum allowed voltage on the power net or the maximum rise on the ground net in volts.

Current Region File Format

The following is the format for a Current Region file:

`LABEL` *`name`* `AREA` *`x1 y1 x2 y2`* `LAYER` *`layer`* `CURRENT` *`current`*

LABEL provides a name for the region, AREA specifies the coordinates of the region LAYER specifies the metal layer that the current sink will be placed on, and CURRENT specifies the current value for that region.

Instance Current File Format

The following is the format for a Instance Current file:

By default current values are in mA.

Macro Power File Format

The following is the format for a Macro Power file:

```
cell1 power_value1
cell2 power_value2
...
```

By default power values are in mW.

Pad Location File Format

The following is the format for a Pad Location file:

```
pad1 x_coordinate y_coordinate
pad2 x_coordinate y_coordinate
...
```

This file can be created by Encounter using the Edit Pad Location menu.

Power Gate File Format

The following is the format for a power gate file:

```
Simplest form
cell cellname supply unswitched_pin_name switched switched_pin_name ron
r_value [idsat idsat_value] [ileak ileak_value]
```

For more complex power gate cells that have multiple switched pins, the cell name is specified first on a line by itself and then the data for the individual power gates is specified.

```
cell cellname
supply pin1 switched pin2 ron r_value idsat idsat_value ileak leak_value
```

Examples

- The following command does a static early rail analysis with a bias voltage of 0.8 volts, a VSS net voltage of 0.0, and a pad location file `dtmfchip_powerplan.pp. It` calculates the power and then displays the IR drop.

  ```
  analyze_early_rail -method static -bias_voltage 0.800 -net_voltage 0.000 \
  -volt_limit 0.08 -calculate_power -pad_location_file dtmfchip_powerplan.pp \
  -display_IR -net VSS
  ```

- The following command specifies that the hierarchy `DTMF_INST/TDSP` currents will be scaled by a factor of 2.

  ```
  analyze_early_rail ... -scale_hierarchy_current DTMF_INST/TDSP scale 2
  ```

- The following command specifies that the hierarchy `DTMF_INST/TDSP` current will be a value if 10.

  ```
  analyze_early_rail ... -scale_hierarchy_current DTMF_INST/TDSP 10
  ```

- The following command points to a powergrid library directory

  ```
  analyze_early_rail ... -power_grid_library ./files/TSDN65LPA128X16M8F_dv.cl
  ```

- The following is an example of an entry in power gate file.

  ```
  CELL HEAD8DMTH SUPPLY VDDG SWITCHED VDD RON 140 IDSAT 1.0 ILEAK 1e-3
  ```

- The following is an example of a pad location file. There are other formats supported. See Encounter or Encounter Power System manuals for details.

```
vss1     770.745000       1127.175000       BA
vss2     757.360000       1127.175000       BA
vss3     745.425000       1127.335000       BA
vss4     732.680000       1127.495000       BA
vss5    1071.360000        221.755000       BA
vss6    1061.920000        221.755000       BA
vss7    1049.625000        221.575000       BA
vss8    1031.815000        221.755000       BA
```

analyze_rail

```
analyze_rail
     [-help]
     [-type {domain | net}]
     [-results_directory directory_name]
     name
```

Runs rail analysis on a net or domain.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `analyze_rail` parameter. For a detailed description of the command and all of its parameters, use the man command `man analyze_rail`.
`name`	Specifies the name of the domain or net that you want to analyze.
`-results_directory` *`directory_name`*	Specifies the name of the directory in which to store the output directories created during the analysis. *Default*: Current working directory.
`-type {domain | net}`	Specifies whether to run rail analysis on a domain or a net. A domain is a list of power or ground nets to analyze together. See set_power_pads on page 933 for more details.

Examples

- The following command performs an analysis for the `vdd` net and places the results directories in the `analysis_out` directory:

  ```
  analyze_rail -type net -results_directory analysis_out vdd
  ```

- The next command performs rail analysis on TDSP power domain and writes the results in current working directory.

  ```
  analyze_rail -type domain -results_directory ./  TDSP
  ```

create_hier_view

```
create_hier_view
     [-help]
     -state_directories dir_list
     -view {static | true_dynamic | worst_dynamic}
     [-abstract_view {true | false}]
     -library_name name
     [-output_dir dir]
     [-block_lef lef_file | -block_gds gds_file]
     [-block_lef lef_file | -gds_port_file gds_port_file]
```

Creates a hierarchical power-grid view.

Parameters

`-abstract_view {true | false}`

If set to `true` an abstract view will be created.

`-block_lef` *`lef_file`*

Specifies the name of the block LEF file. The *`lef_file`* must have all power pin port geometries defined for the hierarchical block.

`-block_gds` *`gds_file`*

In absence of block LEF, you can specify block GDS and port label file to generate hierarchical power-grid view.

`-gds_port_file` *`gds_port_file`*

In absence of block LEF, you can specify block GDS and port label file to generate hierarchical power-grid view.

`-help` Outputs a brief description that includes the type and default information for each `create_hier_view` parameter.

For a detailed description of the command and all of its parameters, use the man command `man create_hier_view`.

`-library_name` *`name`*

Specifies the name of the library that the block power-grid view will be placed in. A `.cl` extension will automatically be added to the name.

`-output_dir` *`dir`* Specifies the output directory that the results will be save in.

`-state_directories` *`dir_list`*

Specifies the location of the state directories that has the analysis output information.

`-view {static | true_dynamic | worst_dynamic}`

Specifies the type of power-grid view that should be created.

`static` Generates accurate and fast-accurate power-grid views for the specified hierarchical partition. It saves the average current distribution inside the power-grid view which can be reused when this power-grid view is included in top-level analysis.

`true_dynamic` Generates accurate detailed dynamic power-grid view for the specified hierarchical partition. It saves the transient behavior of the partition in the power-grid view which is reused when this power-grid view is included in top-level analysis. To generate this view, the analysis must be done with the `set_rail_analysis_mode -save_current_files`" option.

`worst_dynamic`

Generates the worst case dynamic power-grid view for the specified hierarchical partition. To generate this view, the analysis must be done with `set_rail_analysis_mode -save_current_files` option.

Examples

- The following command generates a detailed static power-grid view for the hierarchical block using LEF and static state directories for VDD and VSS nets.

```
create_hier_view -block_lef design.lef -library_name design_static \
    -view static -state_directories { VDD_25c_avg_1 VSS_25c_avg_1 }
```

The `design.lef` must have all power pin port geometries defined for the hierarchical block. The generated static hierarchical view library is `design_static.cl`. The library is in binary format and it stores the block's connectivity to the top-level based on the provided LEF data. It stores the extracted power-grid network and current distribution from the provided state directories.

- The following command generates a detailed dynamic power-grid view for the hierarchical block using LEF and dynamic state directories for VDD and VSS nets.

```
create_hier_view -block_lef design.lef -library_name design_dynamic \
    -view true_dynamic -state_directories { VDD_25c_dynamic_1 \
    VSS_25c_dynamic_1 }
```

run_decap_eco

```
run_decap_eco
     [-help]
     -eco_file file
     -state_directory dir
     [-output_dir dir]
```

Runs a decap ECO file for Encounter.

This command is only used for net based decap optimization (decaps are only added on a net stored in the state directory). To run a decap ECO that considers both power and ground nets simultaneously, you must first run domain based analysis using the `analyze-type domain` command and then source the unified decap ECO using the `set_rail_analysis_mode -decap_eco_file` command to reanalyze the design to understand the effect of the added decaps.

Parameters

`-eco_file` *`file`*	Specifies the eco file name.
`-help`	Outputs a brief description that includes the type and default information for each `run_decap_eco` parameter. For a detailed description of the command and all of its parameters, use the man command `man run_decap_eco`.
`-output_dir` *`dir`*	Specifies the output directory.
`-state_directory` *`dir`*	Specifies the location of the state directory.

Examples

- The following command runs an ECO file called `addcap.eco`:

```
run_decap_eco -state_directory vcc_25C_dynamic_1 -eco_file \
    vcc_25C_dynamic_1/decap_opt/addcap.cmd
```

set_advanced_rail_options

```
set_advanced_rail_options
     [-help]
     [-reset | -vstorm2_include_file_begin file_name]
     [-reset | -vstorm2_include_file_end file_name]
```

Provides the ability to include a file that provides a list of additional `vstorm2` commands that should be used for rail analysis.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_advanced_rail_options` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_advanced_rail_options`.
`-reset`	Resets parameters to their default values.
`-vstorm2_include_file_begin` *`file_name`*	Provides the ability to provide a file that includes `vstorm2` commands that will be run prior to the `analyze` command. This can be used for those VoltageStorm commands that do not have an EPS equivalent.
`-vstorm2_include_file_end` *`file_name`*	Provides the ability to provide a file that includes `vstorm2` commands that will be run after the `analyze` command. This can be used for those VoltageStorm commands that do not have an EPS equivalent.

set_dynamic_rail_simulation

```
set_dynamic_rail_simulation
     [-help]
     [-start value]
     [-stop value]
     [-resolution value]
```

An optional command to set the parameters for dynamic rail analysis. By default, dynamic rail analysis uses the start time, stop time, and resolution from current data file. To analyze fewer clock cycles, you can choose different start and stop times using the `-start` and `-stop` options. It is recommended that at least one or two full cycles are simulated to fully charge and discharge the on-chip capacitance.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_dynamic_rail_simulation` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_dynamic_rail_simulation`.
`-resolution` *value*	By default rail analysis picks up the resolution (or step size) saved in the dynamic current files. To achieve better rail analysis performance, you can increase the step size using this option. However, using larger than two times the step size of current data file is not recommended as it can impact accuracy of dynamic IR drop analysis. *Default*: Picoseconds (ps). You can specify the units with the value.
`-start` *value*	Defines the start time to be used for dynamic rail analysis. This option is used with the `-stop` option and enables you to perform a more detailed analysis in specific time ranges. You can specify the unit with the value. *Default*: The default unit is nanoseconds (ns). The default value is the start time from current data file. The start time in the current data file is usually 0ns

`-stop` *`value`*

Defines the stop time to be used for dynamic rail analysis. This option is used with the `-start` option and enables you to perform a more detailed analysis in specific time ranges. You can specify the unit with the value.

Default: The default unit is nanoseconds (ns). The default value is the end of the current data. If multiple current files are specified and they all have different simulation periods, the largest simulation period is used to decide the stop time.

Examples

- The following command sets the dynamic simulation time from 2ns to 12ns with a resolution of 100ps:

```
set_dynamic_rail_simulation -start 2ns -stop 12ns -resolution 100ps
```

set_net_group

```
set_net_group
     [-help]
     -name group_name
     -type [power | ground]
     -nets {net_list}
```

Specifies the list of either power or ground nets that are isolated but need to be analyzed together as a single net. The results are generated for the new grouped net. The new grouped net name must be one of the net names in the group. To analyze grouped net, user must also specify power pads for the grouped net name.

Parameters

`-help`	Outputs a brief description that includes the type and default information for each `set_net_group` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_net_group`.
`-name` *`group_name`*	Name of the net group. Must be one of the net names in the group.
`-nets` **{***`net_list`***}**	Specifies a list of power or ground nets.
`-type [power | ground]`	Specifies whether it is a power or ground net.

Examples

- The following example groups the power nets that will be analyzed together as a group called VDD.

```
set_net_group -name VDD -type power -nets { VDD VDD1 VDD2 VDD3 }
set_power_pads -net VDD -format xy -file VDD.pp
analyze_rail -type net VDD
```

set_package

```
set_package
     [-help]
     -spice model_file
     -mapping mapping_file
```

Specifies the package model information to be used in power rail analysis.

Parameters

`-help`
: Outputs a brief description that includes the type and default information for each `set_package` parameter.

  For a detailed description of the command and all of its parameters, use the man command `man set_package`.

`-mapping` *`mapping_file`*
: Specifies the name of the package terminal mapping file, which contains the mapping of the terminals of the SPICE subcircuit in the package model file (Specified by the `-spice model_file` option above) to the locations at the grid or top-level pins.

`-spice` *`model_file`*
: Specifies a file containing a SPICE subcircuit describing the package model. This file is generated by Allegro Package SI or other package-modeling products.

Examples

- The following command specifies a package model file called `zlx4ur.cir` and a package mapping file `NTX451.ptmf`:

  ```
  set_package -spice zlx4ur.cir -mapping NTX451.ptmf
  ```

Package Model File (RLC)

The package model file contains a SPICE subcircuit describing a package model. It enables Encounter PS to use package models during full-chip static and dynamic power-grid analysis.

Following is an example SPICE package model file containing a package model with RLC and controlled sources:

```
* BEGIN ANSOFT HEADER
* node 1 W_gnd_20_I_gnd_19
* node 2 W_gnd_20_I_gnd_20
* node 3 W_gnd_22_I_gnd_21
* node 4 W_gnd_22_I_gnd_22
* node 5 W_gnd_24_I_gnd_23
* node 6 W_gnd_24_I_gnd_24
* node 7 W_gnd_26_I_gnd_25
* node 8 W_gnd_26_I_gnd_26
* node 9 W_gnd_27_I_gnd_27
* node 10 W_gnd_29_I_gnd_28
* node 11 W_gnd_29_I_gnd_29
* node 12 W_vdd_20_I_vdd_19
* node 13 W_vdd_20_I_vdd_20
* node 14 W_vdd_22_I_vdd_21
* node 15 W_vdd_22_I_vdd_22
* node 16 W_vdd_24_I_vdd_23
* node 17 W_vdd_24_I_vdd_24
* node 18 W_vdd_26_I_vdd_25
* node 19 W_vdd_26_I_vdd_26
* node 20 W_vdd_27_I_vdd_27
* node 21 W_vdd_29_I_vdd_28
* node 22 W_vdd_29_I_vdd_29
* node 23 W_gnd_20_Sink
* node 24 W_gnd_22_Sink
* node 25 W_gnd_24_Sink
* node 26 W_gnd_26_Sink
* node 27 W_gnd_27_Sink
* node 28 W_gnd_29_Sink
* node 29 W_vdd_20_Sink
* node 30 W_vdd_22_Sink
* node 31 W_vdd_24_Sink
* node 32 W_vdd_26_Sink
* node 33 W_vdd_27_Sink
* node 34 W_vdd_29_Sink
* node 35 Ground_Bias
* .SUBCKT l157-208-rlc 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
+ 24 25 26 27 28 29 30 31 32 33 34 35
C001 23 35 1.38503E-12
C002 24 35 1.92369E-12
C003 25 35 1.09735E-12
```

```
C004 26 35 1.11782E-12
C005 27 35 1.24011E-12
C006 28 35 2.23405E-12
C007 29 35 1.33482E-12
C008 30 35 1.884E-12
C009 31 35 1.09074E-12
C00a 32 35 1.13826E-12
C00b 33 35 1.26783E-12
C00c 34 35 1.5292E-12
C001_002 23 24 1.40828E-14
C001_003 23 25 2.20863E-15
C001_004 23 26 9.86129E-16
C001_005 23 27 5.78913E-16
C001_006 23 28 5.69017E-16
C001_007 23 29 1.07449E-12
C001_008 23 30 7.65697E-15
C001_009 23 31 1.65658E-15
        .
        .
        .
V001 1 36 DC 0
V002 2 37 DC 0
V003 3 38 DC 0
V004 4 39 DC 0
V005 5 40 DC 0
V006 6 41 DC 0
V007 7 42 DC 0
V008 8 43 DC 0
V009 9 44 DC 0
V00a 10 45 DC 0
V00b 11 46 DC 0
V00c 12 47 DC 0
V00d 13 48 DC 0
V00e 14 49 DC 0
V00f 15 50 DC 0
V00g 16 51 DC 0
V00h 17 52 DC 0
V00i 18 53 DC 0
V00j 19 54 DC 0
V00k 20 55 DC 0
V00l 21 56 DC 0
```

```
V00m 22 57 DC 0
L001 36 58 1.74043E-08
L002 37 59 1.74635E-08
K001_002 L001 L002 0.962221
L003 38 60 1.53672E-08
K001_003 L001 L003 0.234447
K002_003 L002 L003 0.235024
L004 39 61 1.53819E-08
K001_004 L001 L004 0.234419
K002_004 L002 L004 0.23498
K003_004 L003 L004 0.948549
L005 40 62 1.50014E-08
K001_005 L001 L005 0.11578
K002_005 L002 L005 0.115597
K003_005 L003 L005 0.213198
K004_005 L004 L005 0.213578
L006 41 63 1.49769E-08
K001_006 L001 L006 0.116214
K002_006 L002 L006 0.116043
K003_006 L003 L006 0.212892
K004_006 L004 L006 0.213269
K005_006 L005 L006 0.940091
L007 42 64 1.52676E-08
K001_007 L001 L007 0.0910913
K002_007 L002 L007 0.0908434
K003_007 L003 L007 0.162235
K004_007 L004 L007 0.162208
K005_007 L005 L007 0.354498
K006_007 L006 L007 0.355793
L008 43 65 1.52384E-08
K001_008 L001 L008 0.0915336
K002_008 L002 L008 0.0912949
K003_008 L003 L008 0.16227
K004_008 L004 L008 0.162248
K005_008 L005 L008 0.353129
K006_008 L006 L008 0.354377
K007_008 L007 L008 0.945321
L009 44 66 1.68684E-08
K001_009 L001 L009 0.0517773
K002_009 L002 L009 0.0513906
K003_009 L003 L009 0.0981259
```

```
K004_009 L004 L009 0.0977762
K005_009 L005 L009 0.187694
K006_009 L006 L009 0.187676
K007_009 L007 L009 0.249808
K008_009 L008 L009 0.250085
L00a 45 67 1.85138E-08
K001_00a L001 L00a 0.0334306
K002_00a L002 L00a 0.0329874
K003_00a L003 L00a 0.0719461
K004_00a L004 L00a 0.0714839
K005_00a L005 L00a 0.141832
K006_00a L006 L00a 0.141545
K007_00a L007 L00a 0.181586
K008_00a L008 L00a 0.181365
K009_00a L009 L00a 0.345425
L00b 46 68 1.84322E-08
K001_00b L001 L00b 0.0338262
K002_00b L002 L00b 0.0333883
K003_00b L003 L00b 0.0722036
K004_00b L004 L00b 0.0717488
K005_00b L005 L00b 0.141736
K006_00b L006 L00b 0.141455
K007_00b L007 L00b 0.181318
K008_00b L008 L00b 0.181103
K009_00b L009 L00b 0.343785
K00a_00b L00a L00b 0.965387
L00c 47 69 1.72003E-08
        .
        .
        .
R001 58 23 0.137729
F001R002 23 58 V002 0.253033
R002 59 23 0.139359
F002R001 23 59 V001 0.250074
R003 60 24 0.155269
F003R004 24 60 V004 0.222103
R004 61 24 0.155542
F004R003 24 61 V003 0.221713
R005 62 25 0.178746
F005R006 25 62 V006 0.220583
R006 63 25 0.177802
```

```
F006R005 25 63 V005 0.221754
R007 64 26 0.166014
F007R008 26 64 V008 0.223098
R008 65 26 0.165238
F008R007 26 65 V007 0.224146
R009 66 27 0.164237
R00a 67 28 0.1504
F00aR00b 28 67 V00b 0.284237
R00b 68 28 0.148034
F00bR00a 28 68 V00a 0.288781
R00c 69 29 0.139614
F00cR00d 29 69 V00d 0.249766
R00d 70 29 0.141127
F00dR00c 29 70 V00c 0.247088
R00e 71 30 0.159795
F00eR00f 30 71 V00f 0.219582
R00f 72 30 0.159914
F00fR00e 30 72 V00e 0.219419
R00g 73 31 0.176776
F00gR00h 31 73 V00h 0.221642
R00h 74 31 0.175737
F00hR00g 31 74 V00g 0.222952
R00i 75 32 0.164128
F00iR00j 32 75 V00j 0.224165
R00j 76 32 0.163176
F00jR00i 32 76 V00i 0.225472
R00k 77 33 0.16394
R00l 78 34 0.157733
F00lR00m 34 78 V00m 0.171123
R00m 79 34 0.152706
F00mR00l 34 79 V00l 0.176756
.ENDS l157-208-rlc
```

Package Model File (coupled analysis)

Below are examples of the `package_model_file` for coupled analysis along with the corresponding terminal mapping file.

Package model file

```
.SUBCKT CKT VDDIN GNDIN GND.in0 VDD.in0
RVDD_1_1 VDDIN    N_1        0.05
```

```
LVDD_1_1 N_1      VDD.in0    2.0n
C_1_1    VDD.in0  GND.in0    1.0p
RGND_1_1 GNDIN    M_1        0.05
LGND_1_1 M_1      GND.in0    2.0n
.ENDS CKT
```

Terminal mapping file

```
VDD.in0 pin VDD.in0
GND.in0 pin GND.in0
GNDIN ground_pin
VDDIN power_pin
```

Package Terminal Mapping File

The format of the package terminal mapping file is as follows:

subckt_terminal_name
subckt_terminal_name pin *toplevel_pin_name*
subckt_terminal_name cell *pad_cell_instance_name*
subckt_terminal_name x,y
subckt_terminal_name [*power_pin*|*ground_pin*]

Mapping File Parameters

subckt_terminal_name

Specifies the name of the terminals of the SPICE subcircuit in the package model file to be mapped to the grid locations or the top-level pins.

Subckt_terminal_name without mapping information indicates that the terminal should be kept floating. *Subckt_terminal_name* specified with the power or ground keyword indicates that the subcircuit terminal should be connected to a virtual power or ground node during power-grid analysis.

pin *toplevel_pin_name*

Specifies the name of the top-level pin to which to connect the SPICE subcircuit terminal.

`cell` *`pad_cell_instance_name`*

Specifies the name of the pad cell instance to which to connect the SPICE subcircuit terminal.

`x,y` Specify the location, in microns, where the specific SPICE subcircuit terminal should be attached.

`power_pin` Specifies the name of the virtual power pin to which to connect the SPICE subcircuit terminal.

`ground_pin` Specifies the name of the virtual ground pin to which to connect the SPICE subcircuit terminal.

The following is an example of a terminal mapping file.

```
VDD.in0 pin VDD.in0
GND.in0 pin GND.in0
GNDIN ground_pin
VDDIN power_pin
```

set_pg_nets

```
set_pg_nets
     [-help]
     -net net_name
     -voltage value
     [-threshold value]
     -tolerance value
     -force
```

Specifies the power grid nets.

Parameters

`-force`	Skips the validity check on power nets.
`-help`	Outputs a brief description that includes the type and default information for each `set_pg_nets` parameter. For a detailed description of the command and all of its parameters, use the man command `man set_pg_nets`.
`-net net_name`	Specifies the name of the power or ground net that you want to analyze.
`-threshold value`	Specifies the minimum allowed voltage on the power net or the maximum rise on the ground net, depending on the setting of the `-voltage` parameter. The `-threshold` parameter is used to verify that the power grid passes the limit check and to set the filters for an IR plot. Optional. `set_pg_nets -threshold` takes precedence over `set_rail_analysis_domain -threshold`.
`-tolerance value`	Specifies the tolerance range of the voltage value. Specify a value from 0 to 1 (not including 1). A value of 0.2 results in a range of “voltage +20%” to “voltage -20%.” *Default*: 0.3.
`-voltage value`	Specifies the voltage on the power rail to be analyzed.

Examples

- The following command specifies a `VDD_TDSPCore` value of 0.9 volts with a threshold of 0.89V and a tolerance of 0.3:

  ```
  set_pg_nets -net VDD_TDSPCore -voltage 0.9 -threshold 0.890 -tolerance 0.3
  ```

set_power_data

```
set_power_data
     [-help]
     [-format {current | ascii | area}]
     [-instance instance_name]
     [-offset offset_value]
     [-power value]
     [-scale factor]
     [-repeat time]
     [-bias_voltage value]
     [{file_list}]
     [-reset]
```

Specifies the format and value of the power data.

Parameters

`-bias_voltage` *`value`*

Specifies the value of the bias voltage.

This parameter must be set if `-format` is set to `ascii` or `area`.

{*`file_list`***}**

Specifies the name of the power data files. The file is optional if `-format area` is specified.

`-format {current | ascii | area}`

Specifies the format of the power data.

The power data format can be `current`, `ascii`, or `area`. The parameter `-bias_voltage` must be set if `area` or `ascii` is specified.

- When using the `current` format option, specify the current file (*.ptiavg) that is generated by the power analysis software within EPS . For example,

  ```
  set_power_data -format current
  static_VDD.ptiavg
  ```

- When using the `ascii` format option, specify the `ascii` instance based power file, generated by PowerMeter or CPE software within EPS. For example,

  ```
  set_power_data -format ascii -bias_voltage
  1.08 powermeter.pwr
  ```

- When using the `area` format option, specify the total power of the chip in Watts. This power is distributed based on cell area. Optionally, you can specify the known power for macro cells using a file, for example,

  ```
  set_power_data -format area -bias_voltage
  1.08 -power 10 cell_power.file
  ```

  where,

  `-bias_voltage` is the voltage at which power was characterized; If the analysis voltage is different, this power will be scaled accordingly.

  `cell_power.file` is in two column format with *`cell_name power_in_watts`*.

`-help`

Outputs a brief description that includes the type and default information for each `set_power_data` parameter.

For a detailed description of the command and all of its parameters, use the man command `man set_power_data`.

`-instance` *`instance_name`*	Specifies the hierarchical instance name for which the current files are included. This option is currently supported only when the power format is set to `current`. This option is useful when power is calculated for hierarchical partitions of the design, enabling you to read power for the hierarchical partitions at the top-level. For example, `set_power_data -format current -instance xy {dynamic_VDD.ptiavg dynamic_VSS.ptiavg}`
`-offset` *`offset_value`*	Specifes the start time offset for current waveforms saved inside the current files. The default unit is in seconds; However, you can also specify the unit with the offset time value. This option is useful in the hierarchical flow to align current waveforms of various partitions. It can be also used to shift power-up rush current waveforms generated by the UltraSim software. For example, `set_power_data -format current -offset 1ns -instance xy {dynamic_VDD.ptiavg dynamic_VSS.ptiavg}`
`-power` *`value`*	Specifies the total power of the chip in Watts and is used with `-format area` option.
`-repeat` *`time`*	Repeats the dynamic current waveform for the specified time. You must ensure that `set_dynamic_rail_simulation -stop` is also specified and it matches the repeat duration. This option is only available for current format files during dynamic analysis. The repeat option can be used when multiple current files have different simulation periods or when IR drop analysis needs to be run for longer duration. See example below. *Default units*: `ns`.
`-reset`	Resets all the previously set power data values.

`-scale` *`factor`*	Specifies a scale factor to apply to the current data. You can use this parameter to scale currents to a more optimistic or pessimistic power-consumption estimate. This parameter is only for `-format current` and `-format ascii`.

Examples

- The following command specifies a format of type current and a scale of 1 for the three given files.:

```
set_power_data -format current -scale 1 {static_VDD_TDSPCore.ptiavg \
static_VDD_TDSPCore_R.ptiavg static_VSS.ptiavg}
```

- The next example specifies two current data files. The first file specifies `vdd_vdd.ptipeak`. The second one specifies `dynamic_VDD.ptiavg` and this current data will be repeated every 100ns, which is the stop time specified for the dynamic analysis.

```
set_power_data -format current vdd_vdd.ptipeak
set_power_data -format current -repeat=100ns dynamic_VDD.ptiavg
set_dynamic_rail_simulation -stop 100ns
analyze_rail ...
```

set_power_pads

```
set_power_pads
     [-help]
     -net net_name
     -format [defpin | padcell | xy | boundary]
     [-file file]
     [-reset]
```

Specifies power pad information.

You can specify multiple power pad formats (xy coordinates, pad cell, or power pin) for the same power net, in both, net-based and domain-based rail analysis

Parameters

`-file` *`file`*
: Specifies the name of the file containing the voltage source information. Not required for `-format defpin` option.

`-format [defpin | padcell | xy | boundary]`
: Specifies how the voltage source is determined.

  Required for `xy`, `padcell`, or `boundary`.

  `defpin`
  : Specifies that voltage sources will be determined by power pins.

  `padcell`
  : Specifies that voltage sources will be determined by pad cells.

  `xy`
  : Specifies that voltage sources will be specified by x y coordinates.

  `boundary`
  : Specifies that voltage source locations will be used for instances in the design.

`-help`
: Outputs a brief description that includes the type and default information for each `set_power_pads` parameter.

  For a detailed description of the command and all of its parameters, use the man command `man set_power_pads`.

`-net` *`net_name`*
: Specifies the name of the power net.

`-reset`
: Resets all the previous `set_power_pads` commands

Examples

- The following command specifies the power pad information for the `VDD_TDSPCore_R` net in the file `dtmf_chip.VDD_TDSPCore_R.vsrc`. The voltages sources are specified by x y coordinates.

```
set_power_pads -net VDD_TDSPCore_R -format xy -file \
dtmf_chip.VDD_TDSPCore_R.vsrc
```

- The following command specifies `xy`, `padcell`, and `defpin` formats for a single net (VDD) as follows:

```
set_power_pads -net VDD -format xy -file VDD.pp
set_power_pads -net VDD -format padcell -file VDD.padcell
set_power_pads -net VDD -format defpin
```

set_rail_analysis_domain

```
set_rail_analysis_domain
     [-help]
     -name domain_name
     -pwrnets power_net_list
     -gndnets ground_net_list
     [-threshold value]
```

Specifies a rail analysis power domain.

Note: For a power-gated domain, you should only specify the “always on” net. The switched net should not be specified as part of power domain definition, because it is traced by the program automatically.

Parameters

`-gndnets` *`ground_net_list`*

Specifies a list of ground nets that are in the domain.

`-help`

Outputs a brief description that includes the type and default information for each `set_rail_analysis_domain` parameter.

For a detailed description of the command and all of its parameters, use the man command `man set_rail_analysis_domain`.

`-name` *`domain_name`*

Specifies the name of the power domain.

`-pwrnets` *`power_net_list`*

Specifies a list of power nets that are in the domain.

`-threshold` *`value`*

Specifies the minimum allowed voltage on the power net or the maximum rise on the ground net, depending on the setting of the `-voltage` parameter. The `-threshold` parameter is used to verify that the power grid passes the limit check and to set the filters for an IR plot.

Optional. `set_pg_nets -threshold` takes precedence over `set_rail_analysis_domain -threshold`.

Examples

- The following command sets the power domain TDSP to include VDD_TDSPCore_R and VSS nets:

  ```
  set_rail_analysis_domain -name TDSP -pwrnets VDD_TDSPCore_R -gndnets VSS
  ```

set_rail_analysis_mode

```
set_rail_analysis_mode
     [-help]
     -method {static | dynamic}
     -accuracy {fast | accurate | fast_accurate}
     -power_grid_library dir_list
     [-analysis_view view]
     [-off_rails net_name_list]
     [-power_switch_eco {true | false}]
     [-em_models file]
     [-default_package_resistor value]
     [-default_package_inductor value]
     [-default_package_capacitor value]
     [-vsrc_search_distance value]
     [-report_msmv_format {true | false}]
     [-generate_movies {true | false}]
     [-generate_path_analysis_waveform {true | false}]
     [-decap_opt_method {removal | feasibility | timing | aggressive]
     [-decap_removal_method {conservative | aggresssive}]
     [-max_leakage value]
     [-generate_decap_eco {true | false}]
     [-create_die_model [two_port | none]]
     [-generate_block_boundary_voltage_file file]
     [-temp_directory_name directory]
     [-save_transient_states {true | false}]
     [-report_via_current_direction {true | false}]
     [-save_current_files {true | false}]
     [-gds_purpose {flipChip | fullChip}]
     [-gds_file file]
     [-gds_top_cell cell_name]
     [-dont_touch_decaps file]
     [-decap_cell_list file]
     [-filler_cell_list file]
     [-decap_eco_file file]
     [-rms_em_analysis {true | false}]
     [-suppress_message {message_id + }]
     [-disable_analysis_types {list of analysis types}]
```

Specifies how the rail analysis will be performed.

Parameters

```
-accuracy {fast | accurate | fast_accurate}
```

Specifies the accuracy mode of analysis.

In static analysis, the accurate mode runs extraction in signoff mode, which takes into account the manufacturing effects and performs power-ground short detection.

Default: `fast`

`fast`	Uses the fastest power-grid available.
`accurate`	Uses the most accurate power-grid available.
`fast_accurate`	Uses the fastest power-grid view available while still maintaining accuracy.

`-analysis_view` *`view`*

Specifies the analysis view.

`-create_die_model [two_port | none]`

If set to `two_port` generates a two port reduced RC chip model with dynamic switching profile. This die-model can be used with package model in Allegro Package Design software to verify whether package and on-chip dynamic profile meets the target impedance across wide frequency spectrum.

`-decap_cell_list` *`file`*

Specifies a file that lists the names of the decap cells that will be used during decap optimization. This option is used when the decap cells are not tagged in the power-grid library.

For dynamic only.

`-decap_eco_file` *`file`*

Specifies the name of the decap ECO file for domain based analysis that will be created after decap analysis is performed. This file is generated in `decap_opt/addcap.cmd` in the state directory.

For dynamic only.

`-decap_opt_method {area | feasibility | timing | removal}`

Specifies the decap optimization method to use.

For dynamic only.

Default: `feasibility`

`removal`	Removes decaps in the design that do not impact the dynamic IR drop threshold set during analysis. A decap removal ECO file will be generated after analysis.
`feasibility`	Provides the ability for inserting decap cells where filler cells have be placed in the design to remove dynamic IR drop violations.
`area`	Provides the ability to add decaps within a specified area of the design to remove IR drop violations. The design is broken up into a grid of fix sized areas and the tool determines how many additional decaps should be added to each of these areas.
`timing`	Provides the ability to focus on instances in a time critical path during decap optimization to ensure that dynamic IR drop will not cause setup or hold time violations while lowering IR-drop constraints at non-critical instances.

`-decap_removal_method` {`conservative` | `aggressive`}

Specifies the guard band used for IR drop threshold when removing decaps.

For dynamic only and used if option: `-decap_opt_method` removal is set.

`conservative`	Uses conservative guard band for IR drop threshold when removing decaps.
`aggressive`	Uses aggressive guard band for IR drop threshold when removing decaps.

`-default_package_capacitor` *`value`*

Specifies the default capacitance value of the package resistance-inductance-capacitance (RLC) model.

`-default_package_inductor` *`value`*

Specifies the default inductance value of the package resistance-inductance-capacitance (RLC) model.

`-default_package_resistor` *`value`*

Specifies the default resistance value of the package resistance-inductance-capacitance (RLC) model.

`-disable_analysis_types {`*`list of analysis types`*`}`

Disables specific analysis types during rail analysis.

For example, `set_rail_analysis_mode -disable_analysis_types { reff vu pi pv }` disables `reff`, `vu`, `pi`, and `pv` analysis types during `analyze_rail`.

`-dont_touch_decaps` *`file`*

Specifies the file that includes a list of decaps that will not be removed.

For dynamic only and used if option `-decap_opt_method removal` is set.

`-em_models` *`file`* Specifies the name of a file that includes the electromigration (em) models.

`-filler_cell_list` *`file`*

Specifies a file that lists the names of the filler cells that will be used during feasibility or timing aware decap optimization. This option is used when the filler cells are not tagged in the power-grid library.

For dynamic only and used if either of the following options are set:
`-decap_opt_method feasibility`
`-decap_opt_method timing`

`-gds_file` *`file`* Specifies the name of the GDS file.

`-gds_map` *`file`* Specifies the name of the GDS mapping file.

`-gds_purpose {flipChip | fullChip}`

Specifies whether the GDS data is for flipchip or full chip.

`-gds_top_cell` *`cell_name`*

Specifies the name of the GDS top cell.

`-generate_block_boundary_voltage_file` *`file_name`*

Writes the voltage of the interface nodes to the block boundary voltage file specified by *file*.

`-generate_decap_eco {true | false}`

Specifies that a decap ECO file should be created.

`-generate_movies {true | false}`

Used to specify that dynamic movies will be created. For dynamic only.

Default: `true`

`-generate_path_analysis_ waveform {true | false}`

Generates a voltage waveform for viewing. For dynamic only.

Default: `false`.

`-max_leakage` *value*

Specifies the maximum leakage current for the design in amps for the decap insertion.

For dynamic only and used if option `-decap_opt_method feasibility` is set.

`-help` Outputs a brief description that includes the type and default information for each `set_rail_analysis_mode` parameter.

For a detailed description of the command and all of its parameters, use the man command `man set_rail_analysis_mode`.

`-method {static | dynamic}`

Specifies whether `static` or `dynamic` analysis will be performed.

Default: `static`

`-off_rails` *net_name_list*

Specifies a list of nets to exclude from analysis. Used in designs with power switches.

`-power_grid_library` *dir_list*

Specifies the name of the power-grid view library directories.

`-power_switch_eco {true | false}`

Specifies that a power switch ECO file should be created.

Default: `false`

`-report_msmv_format {true | false}`

Determines whether the IR drop file used for Encounter Timing System supports multi-supply multi-voltage cells (MSMV).

Default: `true`

`-report_via_current_direction {true | false}`

Determines whether the current direction of vias will be reported.

Default: `false`

`-rms_em_analysis {true | false}`

If set to `true`, performs current density analysis based on Irms current. This options is only available in dynamic analysis (`-method dynamic`).

Default: `false`

`-save_current_files {true | false}`

Determines whether Encounter PS automatically saves current files.

Default: `false`

`-save_transient_states {true | false}`

Determines whether Encounter PS automatically saves the transient node voltage data which is required to generate movies.

Default: `false`

`-suppress_message` **{***`message_id`* + **}**

The messages with the specified message ids will not be output.

Default: No messages are suppressed.

`-temp_directory_name` *`directory`*

Specifies a directory to use for temporary files.

`-vsrc_search_distance` *`value`*

Specifies the search distance in microns for adding voltage sources at power pin locations.

Default: `50`

Examples

- The following command sets the rail analysis mode to `static` and `fast` accuracy, along with some other parameters:

```
set_rail_analysis_mode \
-method static \
-accuracy fast \
-power_grid_library fast_library/fast_allcells.c \
-vsrc_search_distance 50 \
-report_via_current_direction false
```

view_analysis_results

```
view_analysis_results
     [-help]
     [-state_directory dir]
     [-power_db file]
     [-pwr_iv_file file]
     [-gnd_iv_file file]
     [-effective_iv_file file]
     [-enable_violation_browser [true | false]]
     [-display [overlay | clear]]
     [-free_data]
     [-temp_directory dir]
     [-visible_layer {layers}]
     [-plot plot_type
         [ -min_filter value1 -max_filter value2
         | -filter filter_values]]
     [-measure_region x1 y1 x2 y2]
```

Provides a single command for viewing the analysis output graphically.

Note: This TCL command is implemented in free-style, which means that executing an incomplete command invokes the form with specified data:

- `view_analysis_results` will invoke the View Analysis Results form
- `view_analysis_results -power_db power.db` will invoke the form and load power.db

Parameters

`-display [overlay | clear]`

A value of `overlay` specifies that the power-grid data is overlaid with the design. A value of `clear` will remove the display of the power-grid data leaving the design display.

`-effective_iv_file` *`file`*

Specifies the effective instance voltage file to display rail analysis plot `eiv`. The effective voltage for the instance is derived using its dynamic power and ground waveforms. It is the worst voltage in the switching window of the instance. The switching window for the instance comes from TWF or VCD. The effective instance voltage file is generated by default during dynamic analysis of the domain. This file is available under top-level state directory as *`domain`*.iv. E.g. `domain_25c_dynamic_1/domain.iv`

`-enable_violation_browser [true | false]`

When set to `true` it enables the violation browser.

`-filter` *`filter_values`*

Specifies the ranges of values that will be plotted. The syntax for the *`filter_values`* is as follows:

{ { *`on_off1 lower_value1 upper_value1`* **}**

{ *`on_off2 lower_value2 uppervalue2`* **}**

... **}**

The *`on_off`* value is set to 1 if the range is to be displayed and 0 if not. The *`lower_value`* and *`upper_value`* specifies the range that will be displayed or not. See example in Examples section below.

The `-plot` option is required if `-filter` option is specified.

Optional, but can not be not be used with `-min_filter` and `-max_filter` options.

`-free_data`

Frees the loaded analysis state directory or power database from the memory.

`-gnd_iv_file` *`dir`*

Specifies the instance ground voltage file.

Ground Instance Voltage (IV) files include the ground rail voltages of all the instances in the design as calculated by Encounter PS IR drop analysis. This switch and field specifies whether you want to read this file and the name of the file.

`-help`

Outputs a brief description that includes the type and default information for each `view_analysis_results` parameter.

For a detailed description of the command and all of its parameters, use the man command `man view_analysis_results`.

`-measure_region` *`x1 y1 x2 y2`*

Measures tap current in the specified current region defined by x1, y1, and x2 y2 coordinates and prints results in the log file. This can be used to verify that the current regions defined have been assigned by the tool to the correct layer and the correct area.

`-min_filter` *`value1`* `-max_filter` *`value2`*

Specifies the minimum and maximum value of the range that will be plotted. Optional but if specified `-plot` option is required and requires both `-min_filter` and `-max_filter` to be specified. Can not be used if `-filter` option is specified.

`-plot` *`plot_type`*

Plots the specified plot type. See Plot Types on page 947 for a description of each of the plot types.

`-power_db` *`file`*

Specifies the power database file. This file was created by using the `-create_binary_db` option of the `set_power_analysis_mode` command.

`-pwr_iv_file` *`file`*

Specifies the instance power voltage file.

Power Instance Voltage (IV) files include the power rail voltages of all the instances in the design as calculated by Encounter PS IR drop analysis. This switch and field specifies whether you want to read this file and the name of the file.

`-state_directory` *`dir`*

Specifies the state directory where the analysis image files have been saved.

`-temp_directory` *`dir`*

Specifies the temporary directory.

`-visible_layer` **{***`layers`***}**

Specifies which layers should be made visible when plotting. Used in conjunction with the `-plot` option.

Plot Types

Plot Type	Description
`ac`	Available/feasible capacitance (Farad) Displays the feasible decoupling capacitance that can be added in the area. The feasible capacitance in the area is calculated based on the filler cells placement. The program matches the placed filler cells with the decap cells available in the power-grid library and computes the feasible capacitance. The capacitance of the decap cell is stored in the power-grid library.
`cap`	Ggrid capacitance (Farad) Plots the capacitance of the extracted power-grid network.
`dd`	Decoupling capacitance (decap) density (Farad/um^2) Plots decap density of the design. The decap density is the cell internal decoupling capacitance in the unit area (μm^2).
`div`	Differential instance voltage (Volts) - Static Plots instance based VDD-VSS voltage. This is only applicable to static analysis. During static analysis, the instance voltage files for power and ground nets are stored inside the state directories as *`net_name`*.iv. e.g. `domain_25c_avg_1/VDD/VDD.iv` and `domain_25c_avg_1/VSS/VSS.iv`. These files must be specified using `-pwr_iv_file` and `-gnd_iv_file` options respectively to generate differential instance voltage plot.
`dr`	Decoupling capacitance required Reports the decoupling capacitance required to fix dynamic IR-drop violations. This option suggests how much decap needs to be added in a given area to fix the IR-drop violations. The voltage threshold used to determine the IR-drop violations for this analysis is the value specified by the `-threshold` option of the `set_pg_nets` command.

Plot Type	Description
`eiv`	Effective instance voltage (Volts) -dynamic Plots the effective instance voltage file. The effective voltage for the instance is derived using its dynamic power and ground waveforms. It is the worst voltage in the switching window of the instance. The switching window for the instance comes from TWF or VCD. The effective instance voltage file is generated by default during dynamic analysis of the domain. This file is available under top-level state directory as *`domain`*.iv. E.g. `domain_25c_dynamic_1/domain.iv` This file must be specified using the `-effective_iv_file` option.
`er`	Electromigration risk The results are displayed for each segment as 1/mean time to failure, in hours.
`fd`	Filler cell density (1/μm^2) Displays the filler cell density per unit area (μm^2). The plot can be used to analyze the feasibility of adding decap cells by replacing the filler cells and not changing the placement of the logic cells.
`freq`	Frequency domain (Hz) Displays the associated clock frequency of all instances in the design. The instances associated with multiple clock domains are displayed using the fastest clock frequency. This plot can be used to analyze and debug the power distribution in the design.
`giv`	Ground-rail instance voltage (Volts) - static Plots instance based voltage. It is based on the IRdrop (bounce) of the ground network. The instance voltage file saved in the ground state directory must be specified with the `-gnd_iv_file` option. The instance voltage file is saved by default during static analysis under the state directory as *`net_name`*.iv. e.g. `VSS_25c_avg_1/VSS.iv`

Plot Type	Description
`ip`	Instance total power (mW) The total power consumed by each instance is displayed. The total power is the sum of the internal, switching and leakage power of the instance. The power data is read from the power database and can be specified using the `-power_db` option or it is queried directly from memory, if power calculation is run during the session.
`ip_i`	Instance internal power (mW) The power consumed by charging and discharging of the internal interconnect and device capacitances of the instance.
`ip_l`	Instance leakage power (mW) The power consumed by devices when not switching.
`ip_s`	Instance switching power (mW) The power consumed by charging and discharging of the interconnect or output load.
`ir`	IRdrop or voltage drop (Volts) Displays the IRdrop across each extracted power-grid segment in the design.
`load`	Loading capacitance (Farad) Displays the output loading capacitance driven by the instance. This plot can be used to analyze and debug the regions with high switching power.

Plot Type	Description
`pi`	Power-switch (gate) current violation: I/Idsat Displays the ratio of current through the power-switch and the saturation current (Idsat) of the power-switch. This plot can be used to analyze and debug whether the current through power-switches exceeds the saturation current characterized for the power-switch devices. The saturation current and on-resistance of the power-switches are characterized and stored inside the power-grid library of the power-switch cell. A ratio of greater than 1 means that the current requirement of the power-gated block can not be met with placed power-switch instance. Either the power-switch needs to be made larger or power-switch instances need to be added.
`piv`	Power-rail instance voltage (Volts) - static Plots instance based voltage. This is based on the IRdrop on the power network. The instance voltage file saved in the power net's state directory must be specified with the `-pwr_iv_file` option. The instance voltage file is saved by default during static analysis under state directory as *`net_name`*.iv. e.g. `VDD_25c_avg_1/VDD.iv`
`pv`	Voltage drop across power-switch (Volt) Displays the IRdrop across power-switch instances. This plot can be used to analyze and debug the IRdrop inside the power-gated block, e.g. if the IRdrop across power-switches is already high, the IRdrop inside the power-gated block will be much higher. In this case, the power-switch placement will have to be refined in order to resolve the IRdrop problem.

Plot Type	Description
`rc`	Resistor current (Amp) Displays the current across power-grid resistor segments. This plot can be used to analyze and debug how the current flows from the voltage sources into the rest of the design, e.g. high current should flow from pads to the top-level routing level layers and then to the lower level metal layers. This plot can also be used to debug current density violations (`rj`).
`reff`	Total resistance of the instance along the least resistive path Displays the total resistance of the instance along the least resistive path, meaning the resistance of the instance along the path from where it gets the most amount of current. This plot can be used to analyze and debug the power-grid integrity early in the design stage. An instance with high resistance is likely to have high static or dynamic IRdrop depending on the vector that is simulated later in the design cycle. This plot can be also used during decap optimization, since regions with high dynamic IRdrop and high resistance may not be improved upon by adding decaps, because the effectiveness of decoupling capacitance depends on the resistance of the power-grid network. You can also highlight the least resistive path of the instance using the GUI and selecting the instance with left mouse button.
`res`	Displays grid resistance in ohm. This plot can be used to view high resistance regions of the power-grid. It is useful during IRdrop and decap analysis to identify weak power-grid connectivity. The least resistance path of any power-grid segment can be highlighted in GUI using *Enable Grid Resistance Path* check button and selecting any power-grid segment using left mouse button. The detailed report for the least resistance path of the selected segment is printed in the console.

Plot Type	Description
`rj`	Resistor current density, J/Jmax. Displays the current density (current per unit area) violation on each power-grid segment using the ratio of calculated current density and maximum allowed current density. A ratio of greater than 1 means that the current density limit of the segment is violated. The current density limits are supplied by the foundry and can be read during analysis using the `set_rail_analysis_mode -em_models` command.
`slack`	Instance slack (Seconds) Displays worst instance slack. The slack data is queried from either the timing database or if the power database is specified using `-power_db` option, then the worst slack data is read from the power database. When using an external TWF, the slack data is read from the TWF.
`tc`	Tap current (Amp) Displays the current consumed by the instance devices or current taps. During rail analysis, it is based on the power consumption of the instance and is distributed inside the cell. The current distribution inside the cell is based on the extracted device netlist stored inside the power-grid library of the cell. The current distribution is generally based on the device width, length, and spice models. This plot can be used to analyze and debug IRdrop inside the macro.
`td`	Transition density (Hz) Displays the worst transition density (activity * frequency) of the instance. This plot can be used to analyze and debug activity distribution and high power density regions.
`unc`	Unconnected segments Displays power-grid segments that are disconnected from the voltage source (power pads). This plot can be used to debug unusually high IRdrop in the region.

Plot Type	Description
vc	Voltage source current (Amp) Displays current through the voltage sources (power pads). This plot can be used to analyze the current carrying capacity of the power-pad.
vu	Voltage drop across package (Volt) Displays voltage drop across package. If analysis is run with package model attached to the power pads, this plot displays the IRdrop due to package RLCK elements.

Examples

- The following command specifies the appropriate files needed for viewing the instance voltage and power data:

```
view_analysis_results \
    -state_directory analysis/static \
    -power_db powermeter.db \
    -pwr_iv_file PIV.txt \
    -gnd_iv_file GIV.txt \
    -effective_iv_file EIV.txt \
    -display overlay
```

- The next command reads the power database, power.db, creates the specified plots, and enables the violation browser:

```
view_analysis_results
    -power_db power.db \
    -plot [ip | ip_i | ip_s | ip_l | freq | td | slack |load] \
    -enable_violation_browser true
```

- The next command specifies that `ir` data will be plotted and only the layers `M1`, `M2`, and `M3` will be visible.

```
view_analysis_results -plot ir -visible_layer {M1 M2 M3}
```

- The next example specifies that `ir` data within the range of 0 and 10 will be plotted:

```
view_analysis_results -plot ir -min_filter 0 -max_filter 10
```

- The next command specifies that `ir` data will be plotted. There are 8 filter ranges that are all turned on. The first has values from 0 to 1, the next 1 to 2, ... :

```
view_analysis_results -plot ir -filter {{1 0 1} {1 1 2} {1 2 3} {1 3 4} \
    {1 4 5} {1 5 6} {1 6 7} {1 7 8}}
```

- The next command specifies that `ir` data will be plotted, violation browser will not be enabled, temp directory name, display will be overlaid, which layers will be visible, and the ranges of data that will be displayed:

```
view_analysis_results \
    -state_directory VDD_25C_avg_1 \
    -enable_violation_browser false \
    -temp_directory /tmp \
    -display overlay \
    -plot ir \
    -visibleLayer {RD M1 V12 M2 V23 M3 V34 M4 V45 M5 V56 M6} \
    -filter {{1 1.1913 1.2} {1 1.1826 1.1913} {1 1.1739 1.1826} \
    {1 1.1652 1.1739} {1 1.1565 1.1652} {1 1.1478 1.1565} {1 1.1391 1.1478} \
    {1 1.1304 1.1391}}
```

view_dynamic_movie

```
view_dynamic_movie
     [-help]
     [-type [ir|tc] ]
     [-movies_directory dir]
```

Provides the ability to view movies created during dynamic analysis. Movies are sequences of analysis plots displayed over a specific simulation time.

Parameters

`-help`
: Outputs a brief description that includes the type and default information for each `view_dynamic_movie` parameter.

  For a detailed description of the command and all of its parameters, use the man command `man view_dynamic_movie`.

`-movies_directory` *`dir`*
: Specifies the movie directory that was generated during dynamic rail analysis (that is, `state_directory/movie_analysis`)

`-type [ir | tc]`
: Specifies the movie type.

  `ir` IRdrop

  `tc` Tap current

Examples

- The below command views the dynamic movie of type `ir` in the specified state directory:

```
view_dynamic_movie -type ir -movies_directory \
dynamic_rail/TDSP_25C_dynamic_1/VDD_TDSPCore_R/movie_analysis
```

view_dynamic_waveform

```
view_dynamic_waveform
     [-help]
     [-type [current | voltage]]
     [-waveform_files {file1 file2 ... fileN}]
     [-instance_name instance_name]
     [-composite_waveform_type [hierarchy | clock | clock_with_seq
         | total_current]
     [-composite_waveform_name [hierarchy_name | clock_name]]
     [-power_db [powermeter.db]]
     [-effective_voltage_waveform]
     [-free_data]
```

Provides the ability to view dynamic current or voltage waveforms generated during analysis.

Parameters

`-composite_waveform_name [hierarchy_name | clock_name]`

Only available for `-type current`.

If `hierarchy_name` is specified, plots total current waveform for the specified hierarchy.

If `clock_name` is specified, plots current waveform for the selected or all clocks.

If `-composite_waveform_type clock_with_seq` is specified, plots total current waveform for all sequential cells.

If `-composite_waveform_type total_current` option is specified, plots total current waveform for all instances in the current file.

`-composite_waveform_type [hierarchy | clock | clock_with_seq | total_current]`

You can specify to view a waveform for a specific hierarchy, a clock, clock with sequential cells, or total current.

`-effective_voltage_waveform`

Plots the effective voltage waveform (i.e. VDD - VSS).

User must specify the voltage for two nets to generate the effective waveform.

`-free_data` Frees the memory but leaves the menu entries.

`-help` Outputs a brief description that includes the type and default information for each `view_dynamic_waveform` parameter.

For a detailed description of the command and all of its parameters, use the man command `man view_dynamic_waveform`.

`-instance_name` *`instance_name`*

Plot the waveform for the specified instance.

`-power_db [`*`powermeter`*`.db]` Specifies the power database.

Must be specified if `-composite_waveform_type` is `clock` or `clock_with_seq`.

`-type [current | voltage]` Specifies either current (power) or voltage waveform will be displayed.

`-waveform_files {`*`file1 file2 ... fileN`*`}`

Specifies one or more current or voltage waveform files generated with a `.ptiavg` extension.

Examples

- The following command views dynamic composite total current waveforms from the file `vddm_vddm.ptipeak`:

```
view_dynamic_waveform  -type current -composite_waveform_type total_current \
    -waveform_files vddm_vddm.ptipeak
```

- The following command views the dynamic current waveform for instance `inst1`:

```
view_dynamic_waveform \
    -type current \
    -waveform_files dynamic_vss.ptiavg \
    -instance_name inst1
```

- The following command is for viewing the total current waveform:

```
view_dynamic_waveform \
    -type current \
    -waveform_files dynamic_vss.ptiavg \
    -composite_waveform_type total_current
```

- The following command plots a clock domain composite waveform:

```
view_dynamic_waveform \
    -type current \
    -waveform_files dynamic_vss.ptiavg \
    -power_db power.db \
    -composite_waveform_type clock \
    -composite_waveform_name clk
```

- The following command plots a hierarchical composite waveform:

```
view_dynamic_waveform \
    -type current \
    -waveform_files dynamic_vss.ptiavg \
    -composite_waveform_type hierarchy \
    -composite_waveform_name ethernet_mac_2
```

23

Converting EPS (Next-Generation VoltageStorm) Commands and Wrappers

run_libgen

```
run_libgen
     [-help]
     [-check_compatibility list_of_libraries]
     [-cmd cmdfile]
     [-force]
     [[-long_summary | -ls] {library cellname ...}]
     [-report library]
     [[-s | -summary] {library cellname ...}]
     [-techlayers {techfile | library}]
     [-v | -version]
     [-vs_to_eps cmdfile]
```

Executes the libgen wrapper within eps.

Parameters

`-check_compatibility` *`list_of_libraries`*

Checks for compatibility between the specified cell libraries.

`-cmd` *`cmdfile`* Specifies the name of the eps command file.

`-force` If `run_libgen` is invoked with the -force option, it will clean up the tmp and .cl directories and start over, rather than having the user to do the cleanup. This is useful if the first run aborted with errors and there are temp directory and other files hanging around. If libgen is invoked without the -force option, each time that libgen is run, it will create a new pgv.cl (for example, pgv.cl.1 pgv.cl.2 ...).

`-help` Outputs a brief description of the options of this command.

`-long_summary | -ls {library` *`cell_name`*`}`

Generates a detailed summary file.

`library` specifies the name of an existing library and the names of the cells stored in the existing library for which you want to generate a report. If you do not specify a cell name, `LibGen` generates a report for all cells in the existing library.

`-report` *`library`*
: Generates a report from *`library`*.cl with the following contents:

- Cells in the library
- Capacitance data for each cell
- Pass/Fail status for each cell in the library
- Generated power views for each cell in the library
- Number of current taps (devices) extracted for each cell in the library
- Cause and clues for the failed cells in the library

`-s | -summary {library` *`cell_name`*`}`
: Generates a short summary file.

`library` specifies the name of an existing library and the names of the cells stored in the existing library for which you want to generate a report. If you do not specify a cell name, `LibGen` generates a report for all cells in the existing library.

`-techlayers {`*`tech_file | library`*`}`
: The `-techlayers` option takes a technology file, or a cell library name, and prints out all technology layers in top-down order.

`-v | -version`
: Shows the `run_libgen` version.

`-vs_to_eps` *`cmdfile`*
: Specifies the name of the LibGen command file that will be translated to eps.

Examples

- Below is an example for running a libgen with a command file called `libgen.cmd`.

```
run_libgen -cmd libgen.cmd
```

- The below example creates a report of the library `mylib.cl`.

```
run_libgen -report mylib.cl
```

- The below example shows the report generated from the above example.

```
================================================================================
Library path name:            /sev/grp_pe_aedata/MSV_workshop/coupled/libgen
```

```
Library name:                  std_m
Hierarchical library:          NO
User login:                    sdesai
Date:                          2007-Jul-19 14:09:50 (2007-Jul-19 21:09:50 GMT)
ToolIdentifier:                LIBGEN
Application version:           LibGen Version B7.1.0a (07/18/2007)
Number of cells in library:    635
================================================================================
Cell                PowerNets          Capacitance   PowerGrid Views

ADDFHX1M
                    VSS(0.0000)     2.1e-14     Port(1 taps)
                    VDD(0.8000)     2.1e-14     Port(1 taps)
  Cell_Status: PASS
--------------------------------------------------------------------------------|
ADDFHX2M
                    VSS(0.0000)     3.3e-14     Port(1 taps)
                    VDD(0.8000)     3.3e-14     Port(1 taps)
  Cell_Status: PASS
--------------------------------------------------------------------------------|
ADDFHX4M
                    VSS(0.0000)     4.8e-14     Port(1 taps)
                    VDD(0.8000)     4.8e-14     Port(1 taps)
  Cell_Status: PASS
--------------------------------------------------------------------------------|
ADDFHX8M
                    VSS(0.0000)     6.7e-14     Port(1 taps)
                    VDD(0.8000)     6.7e-14     Port(1 taps)
  Cell_Status: PASS
--------------------------------------------------------------------------------|
ADDFHXLM
                    VSS(0.0000)     1.3e-14     Port(1 taps)
                    VDD(0.8000)     1.3e-14     Port(1 taps)
  Cell_Status: PASS
...
...
```

- The following shows the output when using the `-techlayers` option.

```
LibGen - 64-bit Cell Library Preprocessor - Version A6.2.0a (01/14/2007)
--------------------------------------------------------------------------------
              Copyright (c) 2007 Cadence Design Systems, Inc.
```

```
List of layers defined in technology file ./library_dv_pcr.cl/icecaps.tch:
        METAL_5
        VIA_4
        METAL_4
        VIA_3
        METAL_3
        VIA_2
        METAL_2
        VIA_1
        METAL_1
        CONT
        CONT
        CONT
        POLYCIDE
        N_SOURCE_DRAIN
        P_SOURCE_DRAIN
        CSUBSTRATE
```

run_powermeter

```
run_powermeter
     [-help]
     [-cmdlog cmdlog_filename]
     [-dynamic]
     [-logfile filename]
     [tcl_cmd_file]
     [-usage]
     [-v | -version]
     [-vs_to_eps vs_cmdfile]
```

Executes the powermeter wrapper inside `eps`.

Parameters

`-cmdlog` *`cmdlog_filename`*

Creates an additional log file that only has the commands that the user provided. It has no additional output. This log can then be fed back into powermeter "as is" to replay the same commands. The default is `powermeter.cmdlog`.

`-dynamic` — Forces PowerMeter to check out a dynamic license, even if dynamic commands are not part of the command file.

`-help` — Outputs a brief description of the options of this command.

`tcl_cmd_file` — Specifies the name of a EPS command file.

`-usage` — Displays detailed information on command-line parameters but does not start PowerMeter.

`-v | -version` — Shows the `run_powermeter` version.

`-vs_to_eps` *`vs_cmdfile`*

Specifies the name of the PowerMeter command file that will be translated to EPS.

Examples

- The following command runs PowerMeter command file `powermeter.cmd` within eps.

  ```
  run_powermeter powermeter.cmd
  ```

run_vstorm2

```
run_vstorm2
     [-help]
     [-cmd cmd_file]
     [-dynamic]
     [-i]
     [layout_file]
     [-u]
     [-v | -version]
     [-variable value ...]
     [-vs_to_eps vs_cmd_file]
```

Executes the vstorm2 wrapper inside eps.

Parameters

`-cmd` *cmd_file*	Specifies the name of the eps command file.
`-dynamic`	Forces VoltageStorm to check out a dynamic license, even if dynamic commands are not part of the command file.
`-help`	Outputs a brief description of the options of this command.
`-i`	Runs the tool interactively, using the graphical interface.
layout_file	Specifies the name of the GDS or DEF input file.
`-u`	Displays detailed information on command-line parameters but does not start vstorm2.
`-v | -version`	Shows the run_vstorm2 version.
`-variable` *value ...*	Specifies any of the variables listed in Chapter 3, “Environment Variables,” in the *VoltageStorm Cell-Level Rail Analysis Reference Manual*.
`-vs_to_eps` *vs_cmd_file*	Specifies the name of the vstorm2 command file that will be translated to eps.

Examples

The following command runs vstorm2 command file called vstorm2.cmd within eps.

```
run_vstorm2 -cmd vstorm2.cmd
```

vs_to_eps

```
vs_to_eps
     [-help]
     command_file
     -program [libgen | powermeter | vstorm2]
```

Translates `libgen`, `powermeter`, or `vstorm2` command files to `eps` command files.

Parameters

`command_file`	The name of the ANLS command file that will be translated to an Encounter EPS command file.
`-help`	Outputs a brief description of the options of this command.
`-program [libgen | powermeter | vstorm2]`	Specifies which product will be translated to `eps`. The choices are power-grid library generation (`libgen`), power analysis (`powermeter`), or rail analysis (`vstorm2`).

Examples

- The following command translates the libgen command file `libgen_stdcells.cmd` to the EPS command file `libgen_stdcells_eps.tcl`. The program will generate appropriate warnings for the commands or variables that cannot be translated:

  ```
  vs_to_eps -program libgen libgen_stdcells.cmd
  ```

write_anls_command_file

```
write_anls_command_file
     [-help]
     [-help]
     [-generate_only {true | false}]
     [-keep {true | false}]
```

This command is used to output an anls command file.

`-help`	Outputs a brief description that includes the type and default information for each `write_anls_command_file` parameter. For a detailed description of the command and all of its parameters, use the man command `man write_anls_command_file`.
`-keep {true | false}`	If set to `true`, keeps the temporary command file
`-generate_only {true | false}`	If set to `true`, anls binaries are not invoked, only command file is generated.

Examples

- The below command will write a command file in `TMPDIR` and will not invoke `anls` binaries to execute the program:

  ```
  write_anls_command_file -generate_only true
  ```

- The next command will write a command file in working directory and continue to execute the program

  ```
  write_anls_command_file -keep true
  ```

- For the next command, the -keep option will write a command file in working directory and the -generate_only option will not invoke anls binaries to execute the program.

  ```
  write_anls_command_file -generate_only true -keep true
  ```

24

Power-Grid Library Commands

The power-grid library commands are listed below in suggested command order. The commands are in alphabetical order in the chapter.

characterize_power_library

```
characterize_power_library
     [-help]
     [-library_name library_name]
     [-filler_cells cell_name_list]
     [-decap_cells cell_name_list]
     [-default_frequency value]
     [-celllist_file file]
     [-port_powergate {cell supply switched Ron Idsat Ileak}]
     [-detailed_powergate {cell supply switched}]
     [-spice_subckts file_list]
     [-spice_models file_list]
     [-spice_corners corner_list]
     [-gds_files file_list]
     [-gds_layermap file]
     [-current_region_file file]
     [-stop@via layername]
     [-lvsfile_type [xtc | calibre]]
     [-lvsfilename file]
     [-techgen_dir directory]
     [-defaultcap value]
     [-padvsrcfile file]
     [-tablemodels directory_list]
     [-cell_pinnet_map_file file]
     [-cell_accura_data_file file]
     [-libgen_command_file file]
     [-thunder_command_file file]
     [-extraction_command_file file]
     [-output_directory directory]
```

Creates power-grid libraries used for power calculations and analysis.

Parameters

`-cell_accura_data_file` *file*

Specifies the name of a file containing data about the electrical characteristics of cells. Thunder requires this data to generate currents during power-grid view generation.

`-cell_pinnet_map_file` *file*

Specifies the mapping between the cell LEF pin names and the internal names of the net to which those pins are attached. It is used when multiple pins are attached to a single net.

`-celllist_file` *`file`*

Specifies a file containing a list of cells that will be processed.

The `-celllist_file` option is required when `set_power_library_mode -celltype macro` is selected. The program spawns the library generation job in serial mode for each macro in the cell list file. A separate library for each macro is generated. This facilitates debugging when library generation of any macro fails. It also helps overall performance.

`-current_region_file` *`file`*

Specifies a file that includes a list of regions and the amount of current to be distributed within them for the power-grid. This will override any values calculated by Thunder.

`-decap_cells` *`cell_name_list`*

Specifies the names of cells that have explicit decoupling capacitance for use in decap optimization. These cells are nearly equivalent to feedthrough cells, except that there is typically an active device between the power and ground nets to provide extra decoupling capacitance. Power-grid view library generation uses this list of cells to ensure that a capacitance value is associated with each cell. If no capacitance is associated with a cell, a warning message will be issued.

`-default_frequency` *`value`*

Specifies the default frequency of signal nets in hertz (Hz); this is used to calculate the current and capacitance values during the creation of detailed power-grid views.

For example, the following command specifies that the default switching frequency is 100 MHz:

- `characterize_power_library -default_frequency 100e+06`

`-defaultcap` *`value`*

Specifies the default amount of area based decoupling capacitance in a cell.

`-detailed_powergate {`*`cell supply switched`*`}`

Specifies the information that power-grid library generation needs to create a detailed power gate view. Specifies the cell name, the unswitched power net (supply), and the switched power net.

Used in `accurate` mode.

`-extraction_command_file` *`file`*

Specifies the file name of a extraction command file.

`-filler_cells` *`cell_name_list`*

List the names of the filler or feedthrough cells that have no GDS data. These cells can be swapped out for decaps during decap optimization.

`-gds_files` *`file_list`* Specifies the names of the GDSII files used during power-grid library generation.

`-gds_layermap` *`file`* Specifies the mapping of layers between the LEF and GDS files.

`-help` Outputs a brief description that includes the type and default information for each `characterize_power_library` parameter.

For a detailed description of the command and all of its parameters, use the man command `man characterize_power_library`.

`-libgen_command_file` *`file`*

Specifies the file name of a power-grid library generation (libgen) command file

`-library_name` *`library_name`*

Specifies the name of the power-grid library that will be created. A `.cl` extension will be added.

Default: Creates a name based on the parameters of the set_power_library_mode command. i.e. Creates a file named `accurate_stdcells.cl` if `accurate` and `stdcells` were specified.

Note: The parameter honors the output library name that is specified, for all cell types (`set_power_library_mode -celltype`) such as `stdcells`, `allcells`, and `ios`, except `macros`, for which EPS issues an error and generates the library name automatically.

`-lvsfile_type [xtc | calibre]`

Specifies the type of `lvs` input file.

`-lvsfilename` *`file`* Specifies the name of the LVS input file.

`-output_directory` *`directory`*

Specifies the name of the output directory for power-grid library generation.

`-padvsrcfile` *`file`* Specifies the name of a file containing the locations of voltage sources to be connected to pad cells

Used only for cell type IO.

`-port_powergate {`*`cell supply switched Ron Idsat Ileak`*`}`

Specifies the information that power-grid library generation needs to create a port power gate view. Specifies the cell name, the unswitched power pin, the switched power pin, the on resistance in ohms, the saturation current in milliamps, and the leakage current in milliamps.

Used in `fast` mode.

If `ron`, `idsat, or ileak` values are specified, they will be used to override the data extracted by Thunder.

`-spice_corners` *`corner_list`*

Specifies a list of SPICE corners. This is an optional parameter.

If you specify SPICE corner(s), the number of corner items should be equal to the number of SPICE model files specified in `-spice_models` *`file_list`*.

For example, if a SPICE model1 has three corners and model2 has two corners, you should specify:

```
-spice_models { spice_1.model spice_2.model }
-spice_corners { {s11 s12 s13} {s21 s22} }
```

Based on the above parameter description, the following information gets updated in the LibGen file:

```
spice_model_file spice_1.model {s11 s12 s13}
spice_mode_file spice_2.model {s21 s22}
```

`-spice_models` *`file_list`*

Specifies the names of the SPICE model files used by the SPICE netlist file (`-spice_subckts`).

`-spice_subckts` *`file_list`*

Specifies the names of the SPICE netlist files containing the subcircuits corresponding to the cells being processed during power-grid library generation. These subcircuit netlists will be used to calculate pin capacitances using Thunder.

`-stop@via` *`layername`*

Stops the extraction of the power-grid network inside a cell at the specified via layer. This is for quick-view generation.

`-tablemodels` *`directory_list`*

Specifies the location of the set of Thunder model tables that power-grid library generation uses to generate the current data for the cells in the library.

`-techgen_dir` *`directory`*

Specifies the `techgen` directory.

`-thunder_command_file` *`file`*

Specifies the file name of a thunder command file.

Examples

- The following command creates a power-grid library:

```
characterize_power_library \
    -filler_cells {FILLER1 FILLER2 *NFILL8} \
    -decap_cells {DCAP1 DECAP2 NCAP*} \
    -port_powergate {pgate1 vddon vddswitch 1.0 2.0e-03 2.0e-03} \
    -port_powergate {pgate2 vddon vddswitch 2.0 2.0e-03 2.0e-03} \
    -defaultcap 0.100 \
    -libgen_command_file extra_cmd.cmd \
    -output_directory allcells_fastmodel
```

- The following command creates another power-grid library:

```
characterize_power_library \
    -filler_cells FILL* -decap_cells DCAP* \
    -output_directory fast_allcells \
    -defaultcap 0.100 \
    -port_powergate {{HDRSID0 TVDD VDD 673 0.195 2.2e-06}}
```

- The following command creates another power-grid library:

```
characterize_power_library \
    -celllist_file ../DATA/cell_list.mem \
    -spice_models cln65lp_lk_v1d0.l \
    -spice_corners {SS SS_LVT SS_HVT SS_25 SS_DIO SS_DIO_LVT SS_DIO_HVT} \
    -gds_files {tsdn65lpa128x16m8f_100a.gds.gz \
        tsdn65lpa1024x32m8f_100a.gds.gz} \
    -gds_layermap ../DATA/gds.layermap \
    -spice_subckts {mem1.spi mem2.spi} \
    -stop@via CONT \
    -output_directory memories_accurate
```

check_power_library

```
check_power_library
     [-help]
     [-check_compatibility list_of_libraries]
     [-report_file file]
     [-view_cell {all | cell_name}]
     [-stripes dir]
     [-gds file]
     lib_name
```

Generates a report from a power-grid library, *lib_name*.cl with the following contents:

- Cells in the library
- Capacitance data for each cell
- Pass/Fail status for each cell in the library
- Generated power views for each cell in the library
- Number of current taps (devices) extracted for each cell in the library
- Cause and clues for the failed cells in the library

Parameters

`-check_compatibility` *list_of_libraries*

Checks for compatibility between the specified cell libraries.

`-gds` *file*
: Specifies the GDS file.

`-help`
: Outputs a brief description that includes the type and default information for each `check_power_library` parameter.

  For a detailed description of the command and all of its parameters, use the man command `man check_power_library`.

lib_name
: Specifies the absolute path to the power library (`.cl`) file.

`-report_file` *file*
: Specifies the name of the report file that the output will be saved in.

`-stripes` *dir*
: Specifies the stripes directory.

`-view_cell {all |` *cell_name*`}`

Specifies all cells or a cell name.

Examples

- The following command creates a report for the library std_cells.cl:

  ```
  check_power_library std_cells_fast/std_cells_fast.cl.
  ```

- The next command is an example of the report file.

```
================================================================================
Library path name:             /sev/grp_pe_aedata/MSV_workshop/coupled/libgen
Library name:                  std_m
Hierarchical library:          NO
User login:                    sdesai
Date:                          2007-Jul-19 14:09:50 (2007-Jul-19 21:09:50 GMT)
ToolIdentifier:                LIBGEN
Application version:           LibGen Version B7.1.0a (07/18/2007)
Number of cells in library:    635
================================================================================
Cell                PowerNets          Capacitance   PowerGrid Views

ADDFHX1M
                    VSS(0.0000)     2.1e-14     Port(1 taps)
                    VDD(0.8000)     2.1e-14     Port(1 taps)
  Cell_Status: PASS
--------------------------------------------------------------------------------|
ADDFHX2M
                    VSS(0.0000)     3.3e-14     Port(1 taps)
                    VDD(0.8000)     3.3e-14     Port(1 taps)
  Cell_Status: PASS
--------------------------------------------------------------------------------|
ADDFHX4M
                    VSS(0.0000)     4.8e-14     Port(1 taps)
                    VDD(0.8000)     4.8e-14     Port(1 taps)
  Cell_Status: PASS
--------------------------------------------------------------------------------|
ADDFHX8M
                    VSS(0.0000)     6.7e-14     Port(1 taps)
                    VDD(0.8000)     6.7e-14     Port(1 taps)
  Cell_Status: PASS
--------------------------------------------------------------------------------|
ADDFHXLM
                    VSS(0.0000)     1.3e-14     Port(1 taps)
                    VDD(0.8000)     1.3e-14     Port(1 taps)
```

```
  Cell_Status: PASS
...
```

create_ecsm_extension

```
create_ecsm_extension
     [-help]
     -cell_list list
     -spice_subCkt file_list
     [-spice_model {{model1 cornerx cornery} {model2 cornerz}....}]
     -timing_library lib
     [-ecsm_slews slew_table]
     [-ecsm_load load_table]
     [-output_dir dir]
     [-vdd_rail power_rail_name]
     [-gnd_rail ground_rail_name]
```

Enables you to characterize the power and ground current waveforms into an existing Liberty (`.lib`) file.

To analyze the power-grid behavior during dynamic analysis in a more accurate way, the power usage of individual cells attached to the power grid need to be studied. Determining the actual current drawn from the power grid at any given time for any given cell enables the analysis tool to provide a more accurate picture of the general power grid behavior. The `create_ecsm_extension` command enables you to characterize the power and ground current waveforms into an existing Liberty (`.lib`) file. The existing power and timing data in the `.lib` file are retained and only the Power Dynamic Current (PDC) and Ground Dynamic Current (GDC) waveforms are added corresponding to the values in the power table for each arc.

The characterization tool, `create_ecsm_extension` command measures and generates this data for standard cells and obtains the current profiles for the supply and ground pins for all transitioning arcs of the cell, under different input slew and output load combinations, and under different sensitizations of side input. The power/ground dynamic current waveforms in the `.lib` have SPICE level accuracy and read in by power analysis for dynamic current calculation. PowerMeter interpolates the waveforms if loads and slews do not match the index values of `.lib`. The dynamic current files outputted by PowerMeter is then used in VoltageStorm® Dynamic Gate (VSDG) for dynamic IR drop analysis.

Parameters

`-cell_list` *`list`*
: Specifies a list of cells.

`-ecsm_load` *`load_table`*
: Specifies the load table

`-ecsm_slews` *`slew_table`*

Specifies the ECSM slew table.

`-gnd_rail` Specifies the name of the ground rail.

`-help` Outputs a brief description that includes the type and default information for each `gcreate_escm_extension` parameter.

For a detailed description of the command and all of its parameters, use the man command `man create_escm_extension`.

`-output_dir` *`dir`* Specifies the output directory

`-spice_model` **{{***`model1 cornerx cornery`***} {***`model2 cornerz`*`}....`**}**

Specifies files with SPICE models and corners.

`-spice_subckt` *`file_list`*

Specifies the SPICE subcircuit files.

`-timing_library` *`lib`*

Specifies the timing library.

`-vdd_rail` Specifies the name of the power rail.

Examples

- The following command characterizes the power and ground current waveforms into the existing Liberty(`.lib`) file:

```
create_ecsm_extension \
-cell_list cell_list \
-spice_subckt cln65lp_lk_v1d0.l \
-spice_model { { nvt.spi SS MID_DIO } { mod.typ} } \
-timing_library LIB/tcbn65lp_c050428wc.lib \
-vdd_rail VDD \
-gnd_rail VSS \
-ecsm_slews {0.010 0.025 0.050 0.100 0.200 0.500 1.00} \
-ecsm_load {0.010 0.025 0.050 0.100 0.200 0.500 1.00} \
-output_dir dpm_nvt
```

set_power_library_mode

```
set_power_library_mode
     [-help]
     -accuracy {fast | accurate}
     -celltype {allcells | stdcells | macros | ios }
     [-input_type {pr_lef | pr_lefgds | pr_gds | pr_techonly}]
     -extraction_tech_file file
     -lef_layermap file
     -generic_power_names pin1 voltage1 ... pinN voltageN
     -generic_ground_names ground_pin_list
     [-mbb_layer layer_name]
```

Specifies what kind of power-grid library will be created.

Parameters

`-accuracy {fast | accurate}`

Determines what power-grid views will be created, dependent on whether fast or accurate power analysis is to be done.

Default: `fast`

Note: The `fast` accuracy option supports cell characterization only for `allcells` cell type.

The `accurate` accuracy option does not support `allcells` cell type.

`-celltype {allcells | stdcells | macros | ios}`

Specifies the cell type used for library creation.

- `allcells`

  Generates port views for all the cells available in the LEFs, only if you specified the `-accuracy fast` power analysis option. This library is useful for pipe-cleaning designs.

- `stdcell`

  Generates a single library for all the standard cells using spice netlist or extraction/gds netlist, only if you specified the `-accuracy accurate` power analysis option.

- `io`

  Generates a single library for I/O's with spice netlist/gds or xtc/gds, only if you specified the `-accuracy accurate` power analysis option.

- `macro`

  Generates libraries for macro cells using spice netlist/gds and xtc/gds, only if you specified the `-accuracy accurate` power analysis option.

`-extraction_tech_file` *`file`*

Specifies the name of the technology file that will be used for extraction.

`-generic_ground_names` *`ground_pin_list`*

Specifies the names of the nets considered ground nets for all cells for which the ground nets are not specified.

`-generic_power_names` *`pin1 voltage1 ... pinN voltageN`*

Specifies the names of the nets considered power nets for all cells for which the power nets are not specified.

`-help`

Outputs a brief description that includes the type and default information for each `set_power_library_mode` parameter.

For a detailed description of the command and all of its parameters, use the man command `man set_power_library_mode`.

`-input_type {pr_lef | pr_gds | pr_lefgds | pr_techonly}`

Specifies the type of input that will be used for generating power-grid libraries. For `pr_lef` the pin information comes from the LEF file. For `pr_gds` flow there is only a technology LEF and a GDS file, so the LEF information must be created from the GDS information. For `pr_lefgds` some of the ports are defined in the LEF and the remaining ones will be created from the GDS file.

Specify `-input_type pr_techonly` to create a cell library with technology only information without any physical or electrical data. The `set_power_library_mode` command issues error if you dont specify the LEF file (`read_lib -lef`), the extraction technology file (`-extraction_tech_file`) or the mapping layers between the LEF and technology file (`-lef_layermap`).

Note: For power-grid library generation, the `set_power_library_mode` command uses that output directory specified in `characterize_power_library -output_directory`. If not, the command uses the current working directory to generate the output. Similary, the command uses the library name specified in `characterize_power_library -library_name`; If not, the command uses techonly_library as the default library name.

Default: `pr_lef`

For information on setting the input type using the GUI, see the *Set Power Library Mode* form in Power and Rail Menu.

`-lef_layermap` *`file`*

Determines that mapping of layers between the LEF and technology file.

`-mbb_layer` *`layer_name`*

Identifies the layer name that library generation uses to compute the placement bounding box for GDSII cells that are placed by the DEF file. Enabled if input type is `pr_gds`.

Examples

- The following command sets the power library mode:

```
set_power_library_mode \
    -accuracy fast \
    -celltype allcells \
    -input_type pr_lef \
    -extraction_tech_file RCgen.tch \
    -lef_layermap lefdef.map \
    -generic_power_names {VDD 1.2 DVDD 1.2 AVDD 3.3} \
    -generic_ground_names {VSS GND AVSS}
```

- The next command also sets the power library mode:

```
set_power_library_mode \
    -accuracy fast -celltype allcells \
    -extraction_tech_file cln65lp_1p08m+alrdl_typical.tch \
    -lef_layermap lefdef.layermap \
    -generic_power_names {VDDm 0.84 VDD 0.84 TVDD 0.84 VDDL 0.84} \
    -generic_ground_names VSS \
    -input_type pr_lef
```

25

Power Planning Commands

addRing

```
addRing
     -nets {list_of_nets}
     -type
          {core_rings
               -follow {core | io}
               [-exclude_selected {0 | 1}]
               [-around user_defined ]
          | block_rings -around
               {each_block
               | each_reef
               | power_domain
               | selected
               | cluster
               | shared_cluster
               | user_defined}}
     [-user_defined_region {coordinate_list}]
     -layer_top layer
     -layer_bottom layer
     -layer_left layer
     -layer_right layer
     -width_top width
     -width_bottom width
     -width_left width
     -width_right width
     -spacing_top spacing
     -spacing_bottom spacing
     -spacing_left spacing
     -spacing_right spacing
     {-center {0 | 1} | -offset_top offset -offset_bottom offset
          -offset_left offset -offset_right offset}
     [-offset_adjustment {automatic | fixed}]
     [-tl {0 | 1}] [-tr {0 | 1}]
     [-bl {0 | 1}] [-br {0 | 1}]
     [-lt {0 | 1}] [-rt {0 | 1}]
     [-lb {0 | 1}] [-rb {0 | 1}]
     [-top {0 | 1}] [-bottom {0 | 1}] [-left {0 | 1}] [-right {0 | 1}]
     [-rectangle {0 | 1}]
     [-use_wire_group {0 | 1}]
     [-use_wire_group_reinforcement_group_via {0 | 1}]
     [-use_interleaving_wire_group {0 | 1}]
     [-use_wire_group_bits integer]
     [-threshold {value | auto}]
     [-jog_distance value]
     [-snap_wire_center_to_grid {None | Grid | Half_Grid | Either}]
     [-stacked_via_top_layer layername]
     [-stacked_via_bottom_layer layername]
     [-via_using_exact_crossover_size {0 | 1}]
     [-orthogonal_only [0 | 1]]
     [-split_long_via {threshold_value step_value offset_value}
```

```
[-skip_via_on_wire_shape { [Blockring] [Stripe] [Padring] [Ring] [Noshape]}]
[-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}]
[-max_via_size {shape width% height% target_penetration%}]
```

Creates rings for specified nets around the core boundary or selected blocks and groups of core rows. Use this command after creating an initial floorplan.

Parameters

`-around`
: Specifies the objects around which to create rings. This parameter is used in conjunction with the `-type` parameter. Specify one of the following:

  `each_block`
  : Creates rings around each block in the design. This option can only be used if you also specify `-type block_rings`.

  `each_reef`
  : Creates a separate ring around each reef in the design. Reefs are used in flipchip designs.

  `power_domain`
  : Creates rings around each power domain or around each fence that surrounds a soft block in the design. This option can only be used if you also specify `-type block_rings`.

  `selected`
  : Creates rings around each block and each group of core rows that is selected in the design display area. This option can only be used if you also specify `-type block_rings`.

  `cluster`
  : Treats all the blocks and groups of core rows selected in the design display window as a single cluster and creates rings around the cluster. This option can only be used if you also specify `-type block_rings`.

  `shared_cluster`
  : Creates rings around all selected blocks with shared edges residing in the middle of the common areas between adjacent blocks. This option can only be used if you also specify `-type block_rings`.

`user_defined`

Creates a rectilinear ring using the specified the coordinates. This option can be used with either `-type core_rings` or `-type block_rings`. If you specify `user_defined`, you must also specify the `-user_defined_region` parameter.

Note: You must specify the coordinates in a linear sequence. For example, you cannot specify the coordinates in a sequence such as bottom left, top right, bottom right, top left.

`-bl {0 | 1}`

If set to `1`, extends the bottom segment of the ring to the left until encountering a target.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the bottom segment of the ring is not extended to the left.

Note: If you use the ring extension functionality for block-level designs, the software automatically creates pins at block edges.

`-br {0 | 1}`

If set to `1`, extends the bottom segment of the ring to the right until encountering a target.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the bottom segment of the ring is not extended to the right.

Note: If you use the ring extension functionality for block-level designs, the software automatically creates pins at block edges.

`-center {0 | 1}`

Specifies whether to center the core rings between the I/O pads and core boundaries. If you do not specify this parameter with a value of 1, you must specify all of the following parameters:

- `-offset_top`
- `-offset_bottom`
- `-offset_left`
- `-offset_right`

Default: `0`

`-exclude_selected {0 | 1}`

Specifies whether the core ring should be created so that selected objects (blocks and power domains) are not included within its boundary.

Default: `0`

`-follow {core | io}`

Specifies whether core rings are placed along the core boundary or the I/O boundary. This parameter is used only with the `-type core` parameter.

Default: `core`

`-io_offset{0 | 1}` Specifies whether the offset for core rings is measured from the I/O boundary. This parameter is used only with the `-type core` parameter and is mutually exclusive with the `-around` core parameter.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the offset for the core ring is measured in relation to the core boundary.

`-jog_distance` *`value`*

Specifies the least amount of jog allowed (to follow the contour of the referenced object) before a jog is removed. This value is in micrometers. If the ring would need to jog a distance equal to or less than this value in order to follow the contour of the object, the ring will not jog.

Default: If you do not specify this parameter, the jog distance value is automatically calculated based on the worst spacing in the technology file.

`-layer_top` *`layer`* `-layer_bottom` *`layer`*
`-layer_left` *`layer`* `-layer_right` *`layer`*

Specifies which layer to use for each side of the ring or rings being created.

`-lb {0 | 1}`

If set to `1`, extends the left segment of the ring to the bottom until encountering a target.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the left segment of the ring is not extended to the bottom.

Note: If you use the ring extension functionality for block-level designs, the software automatically creates pins at block edges.

`-lt {0 | 1}`

If set to `1`, extends the left segment of the ring to the top until encountering a target.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the left segment of the ring is not extended to the top.

Note: If you use the ring extension functionality for block-level designs, the software automatically creates pins at block edges.

`-max_via_size {`*`shape width% height% target_penetration%`*`}`

The maximum size of a crossover via, specified as a percentage of a full-size via. You must specify three values for each shape that you specify.

The shape values are:

- Pad
- Ring
- Stripe
- Blockring
- Blockpin
- Cover
- Noshape

The three values you must specify for each shape are:

- The maximum width for the crossover via, as a percentage of a full-size via
- The maximum height for the crossover via, as a percentage of a full-size via
- The target penetration, which specifies a smaller size for vias connecting to targets of the specified shape.

`-nets {`*`list_of_nets`*`}`

Specifies the names of the nets for which power rings are to be created. You must enclose the list of net names within curly braces.

`-offset_adjustment {automatic | fixed}`

Specifies whether the offset is automatically adjusted when you change the width or spacing values.

Default: `automatic`

`-offset_top` *`offset`* `-offset_bottom` *`offset`*
`-offset_left` *`offset`* `-offset_right` *`offset`*

Specifies the spacing from the edge of the inner ring to the boundary of the referenced object for each side of the ring. For core rings, the referenced object is either the I/O boundary or the core boundary. For block rings, the referenced object is the block boundary. These parameters are ignored if you use the `-type core_rings -center` option.

`-orthogonal_only [0 | 1]`

Creates vias *only* for orthogonal wires and pin intersections.

Default: If you do not specify this parameter or if you specify it with a value of `1`, the software creates vias for orthogonal wires and pin intersections.

`-rb {0 | 1}`

If set to `1`, extends the right segment of the ring to the bottom until encountering a target.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the right segment of the ring is not extended to the bottom.

Note: If you use the ring extension functionality for block-level designs, the software automatically creates pins at block edges.

`-rectangle {0 | 1}`

If set to `1`, creates rectangular rings only.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the creation of rectilinear rings is permitted.

`-rt {0 | 1}`

If set to `1`, extends the right segment of the ring to the top until encountering a target.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the right segment of the ring is not extended to the top.

Note: If you use the ring extension functionality for block-level designs, the software automatically creates pins at block edges.

`-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}`

Prevents vias from being generated on the specified types of pins.

If you specify `-skip_via_on_pin {}`, the power planning software makes connections to all four categories of pins.

Default: If you do not specify this parameter, vias are generated for block pins, pad pins, and cover macro pins, but are not generated for standard cell pins.

`-skip_via_on_wire_shape {[Blockring] [Stripe] [Padring] [Ring] [Noshape]}`

Prevents vias from being generated on the specified wire shapes.

If you specify `-skip_via_on_wire_shape {}`, the power planning software makes connections to all five categories of wire shapes.

Default: If you do not specify this parameter, vias can be generated on all wire shapes *except* no-shape wires.

`-snap_wire_center_to_grid {None | Grid | Half_Grid | Either}`

Controls the snapping of the center of the rings. If you do not specify this parameter, rings are not snapped to the routing grid. Specify one of the following values:

`None`	Does no snapping of the wires.
`Grid`	Snaps the centers of the wires to the routing grid of the same layer as the wire.
`Half_Grid`	Snaps the centers of the wires to the half-routing grid. The half-routing grid is a virtual grid with the tracks at the center of every two adjacent tracks of the routing grid.
`Either`	Snaps the centers of the wires to the closer of the routing grid or half-grid track.

`-spacing_top` *`spacing`* `-spacing_bottom` *`spacing`*
`-spacing_left` *`spacing`* `-spacing_right` *`spacing`*

Specifies the edge-to-edge spacing between rings for each side of the ring.

`-split_long_via {`*`threshold_value step_value offset_value`*`}`

Splits vias that are longer than the specified length into smaller vias that have the specified center-to-center step and bottom/ left edge offset (all values in micrometers).

Default: If you do not specify this parameter, the software does not split vias.

`-stacked_via_bottom_layer` *`layername`*

Specifies the lowest layer in which vias can be stacked.

Default: The lowest metal layer in the design.

`-stacked_via_top_layer` *`layername`*

Specifies the highest layer in which vias can be stacked.

Default: The highest metal layer in the design.

`-threshold {`*`value`* `| auto}`

`value`

Specifies the least amount of spacing allowed between ring segments of adjacent blocks before the rings are merged. If the spacing is less than this value, the common sides of the rings are merged. Units in micrometers.

`auto`

Specifies that Encounter will automatically determine where to merge the wires being created with the prerouted wires.

Default: 10 μm (if you do not specify this parameter, rings farther than 10 μm from each other are not merged).

`-tl {0 | 1}`

If set to `1`, extends the top segment of the ring to the left until encountering a target.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the top segment of the ring is not extended to the left.

Note: If you use the ring extension functionality for block-level designs, the software automatically creates pins at block edges.

`-top {0 | 1} -bottom {0 | 1} -left {0 | 1} -right {0 | 1}`

Specifies the side or sides of the ring to be created. To prevent a side from being created, use the parameter with a value of `0`.

Default: If you do not specify these parameters or if you specify them with values of `1`, all sides are created.

`-tr {0 | 1}`

If set to `1`, extends the top segment of the ring to the right until encountering a target.

Default: If you do not specify this parameter or if you specify it with a value of `0`, the top segment of the ring is not extended to the right.

Note: If you use the ring extension functionality for block-level designs, the software automatically creates pins at block edges.

`-type {core_rings | block_rings}`

Specifies the ring type to create.

Default: If you do not specify this parameter, core rings are created.

Specify one of the following values:

`core_rings` Creates core rings that follow the contour of the core boundary or the I/O boundary. If you specify `core_rings`, then you must specify either the `-follow` parameter or the `-around user_defined` parameter.

Default: The ring is offset from the core, based on offset values that you specify in the `-offset` parameters.

`block_rings`

Creates block rings that follow the contour of the specified objects. If you specify `block_rings`, then you must specify the `-around` argument with one of its required options.

`-user_defined_region {`*`x1 y1 x2 ya …`*`}`

Specifies the coordinates for a rectilinear region if you use the `-around user_defined` parameter. You must specify at least four sets of coordinates and the coordinate list must be enclosed within curly braces. You must also specify the coordinates in a linear sequence. For example, you cannot specify the coordinates in a sequence such as bottom left, top right, bottom right, top left.

`-use_interleaving_wire_group {0 | 1}`

If set to `1`, connects multiple wires from the same net together, even if separated by wires from a different net. You can only specify this parameter if you also specify `-use_wire_group 1`.

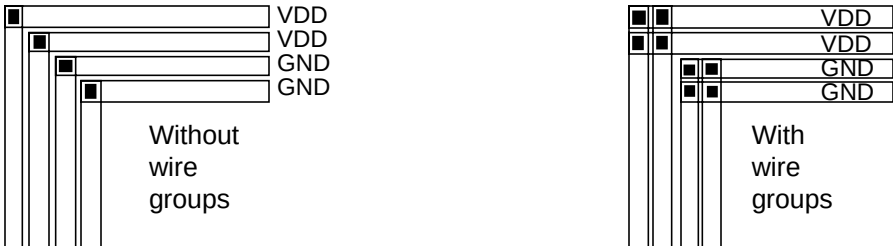
Default: If you do not specify this parameter or if you specify it with a value of `0`, interleaving wire groups are not used.

`-use_wire_group {0 | 1}`

If set to `1`, connects multiple wires from the same net together.

Note: For stripes to connect to rings using wire groups, you must specify `-useWireGroup` in both the `addRing` and `addStripe` commands.

The following figure shows the difference in the connections.



Default: If you do not specify this parameter or if you specify it with a value of `0`, wire groups are not used.

`-use_wire_group_bits` *`integer`*

Specifies how many nets are to be included in the wire group. You can only specify this parameter if you also specify `-use_wire_group 1`.

Default: The same number of nets as specified in the `-nets` parameter.

`-use_wire_group_reinforcement_group_via {0 | 1}`

Creates group vias for wire group corners along with reinforcement stripes placed in the horizontal and vertical directions for the same wires. You can use this parameter to improve the yield and reliability issues associated with vias and reinforcement stripes by ensuring that if one of the vias or reinforcement stripes produces a void, the contact will be maintained through one of the adjoining vias or reinforcement stripes.

Default: `0`

`-via_using_exact_crossover_size {0 | 1}`

Specifies whether to generate partial vias of the exact crossover size. A value of `1` allows partial vias to be generated. A value of `0` prevents partial vias, and the software always generates full-size vias.

Note: This option does not apply when both of the objects at the crossover are rings or stripes, or at the crossover of a ring and stripe. In these cases, the software always generates a full-size via array.

Default: `1`

`-width_top` *`width`* `-width_bottom` *`width`* `-width_left` *`width`* `-width_right` *`width`*

Specifies the width of the ring segments for each side of the ring.

Example

The following command adds a core ring to the vdd and gnd nets on layers M5 and M6:

```
addRing -nets {vdd gnd} -type core_rings -center 1 -layer_top M6 -layer_bottom M6
-layer_right M5 -layer_left M5 -width_top 10 -width_bottom 10 -width_left 10
-width_right 10 -spacing_top 2 -spacing_bottom 2 -spacing_right 2 -spacing_left 2
```

This ring is centered between the I/O pad area and the core boundary, has a width of 10 μm, and is located at least 2 μm away from any other ring.

addSpecialRoute

`addSpecialRoute` *`filename`*

Incrementally adds special route information from a file. This includes only special route wires and vias. If the specified filename is compressed (with the `.gz` extension), the file is read in directly. Use this command after you have imported a design.

Parameters

`filename`	Specifies the name of the file that contains special route information, to be used as input for the design. Only `.spr` files are supported.

addStripe

```
addStripe
     -nets {list_of_nets}
     -layer layer
     [-direction {horizontal | vertical}]
     -width width
     -spacing spacing
     {-set_to_set_distance real | -number_of_sets integer |
          {-over_bumps {0 | 1} | -between_bumps {0 | 1}} | -over_pins {0 | 1}
          -pin_layer layer {-master master_cell | -all_blocks}}
     [-extend_to {first_padring | last_padring | design_boundary | all_domains}]
     [-over_power_domain {0 | 1}]
     [-start_from {left | right}]
     [-start_from {bottom | top}]
     [-create_pins {0 | 1}]
     [-area {x1 y1 x2 y2...}]
     [-area_blockage {x1 y1 x2 y2...}]
     [-xleft_offset value] [-xright_offset value]
          [-ytop_offset value] [-ybottom_offset value]
     | [-start_x value] [-start_y value] [-stop_x value] [-stop_y value]
     [-break_stripes_at_block_rings {0 | 1}]
     [-break_at_selected_blocks {0 | 1}]
     [-switch_layer_over_obs {0 | 1}
     [-allow_jog_padcore_ring {0 | 1}]
     [-allow_jog_block_ring {0 | 1}]
     [-merge_stripes_value value]
     [-max_pin_width value]
     [-max_same_layer_jog_length value]
     [-padcore_ring_top_layer_limit value]
     [-padcore_ring_bottom_layer_limit value]
     [-pin_offset]
     [-block_ring_top_layer_limit value]
     [-block_ring_bottom_layer_limit value]
     [-use_wire_group {0 | 1}]
     [-use_interleaving_wire_group {0 | 1}]
     [-use_wire_group_bits integer]
     [-snap_wire_center_to_grid {None | Grid | Half_Grid | Either}]
     [-stacked_via_top_layer integer]
     [-stacked_via_bottom_layer integer]
     [-via_using_exact_crossover_size {0 | 1}]
     [-orthogonal_only [0 | 1]
     [-split_long_via {threshold_value step_value offset_value}
     [-split_vias {0 | 1}
     [-same_sized_stack_vias {0 | 1}]
     [-same_layer_target_only {0 | 1}]
     [-skip_via_on_wire_shape {[Blockring] [Stripe] [Padring] [Ring] [Noshape]}]
     [-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}]
     [-max_via_size {shape width% height% target_penetration%}]
```

Creates power stripes within the specified area. If the router encounters an obstruction, the stripe connects to the last stripe on the same net. Otherwise, the stripes stop at the core row boundary. Use this command after creating an initial floorplan.

Parameters

`-all_blocks {0 | 1}`

Specifies that stripes should be generated over all blocks.

Default: `0`

`-allow_jog_padcore_ring {0 | 1}`

If set to `0`, stripes cannot jog to connect to the pad ring or core ring. If set to `1`, stripes can jog to connect to the pad ring or core ring.

Default: `1`

`-allow_jog_block_ring {0 | 1}`

If set to `0`, stripes cannot jog to connect to a block ring. If set to `1`, stripes can jog to connect to a block ring.

Default: `1`

`-area {`*`x1 y1 x2 y2...xn yn`*`}`

Specifies the coordinates of the design area in which the stripes must be created. The `-xleft_offset` and `-xright_offset` values are measured from this boundary area for vertical stripes, and the `-ytop_offset` and `-ybottom_offset` values are measured from this boundary area for horizontal stripes.

Note: If a core ring exists in the design, the stripes automatically start and stop at the core ring if you do not use the `-area` parameter.

You can also create stripes in an area defined by a polygon (rectilinear). For example:

```
addStripe -area {100 100 100 200 150 200 150 150 200 150
200 100}
```

Default: If you do not specify this parameter, stripes are created for the entire design area.

`-area_blockage {`*`x1 y1 x2 y2...xn yn`*`}`

Specifies a rectilinear area blockage. Specify the blockage coordinates to restrict the addition of any stripes to this region. This parameter supports multiple coordinates.

`-between_bumps {0 | 1}`

Specifies that stripes should be generated between bumps.

Default: `0`

`-block_ring_bottom_layer_limit` *`layer_name`*

Specifies the lowest layer that stripes can switch to when encountering a block ring. Stripes that encounter a block ring can only switch to this layer or higher.

`-block_ring_top_layer_limit` *`layer_name`*

Specifies the highest layer that stripes can switch to when encountering a block ring. Stripes that encounter a block ring can only switch to this layer or lower.

`-break_at_selected_blocks {0 | 1}`

If set to 1, the stripe stops at the block boundary when the block does not contain a blockage. The stripe stops at the block boundary when the block contains a blockage, and spacing from the block boundary to the edge of the blockage is greater than the minimum spacing value of that layer. The stripe stops at the minimum spacing away from the blockage when the spacing from the block boundary to the edge of the blockage is less than the minimum spacing of that layer.

Default: `0` (If you do not specify this parameter, stripes do not break at selected blocks.)

`-break_stripes_at_block_rings {0 | 1}`

If set to `1`, stripes break and connect to the block rings and ring pins if they are encountered.

Note: This parameter applies only to block rings that surround a block, and does not apply to block rings that surround rows or standard cells. If the block is a ring macro, the stripe will break at the ring pin inside the block ring.

Default: If you do not specify this parameter, stripes do not break at block rings.

`-create_pins {0 | 1}`

If set to `1`, IO pins are created for each stripe. Any overlap with existing IO pins at the design boundary is flagged as a warning after the stripes are created. If set to `0`, pins are not created.

Note: This parameter is only available if you also specify the `-area` parameter.

`-direction {horizontal | vertical}`

Sets the stripe direction.

Default: If you do not specify this parameter, vertical stripes are created.

`-extend_to` The boundary for the stripes. Specify one of the following values:

`all_domains`

Stripes are created over all power domains for each power and ground net within each power domain. Stripes start and stop at each power domain boundary or ring around the power domain, and ignore the values in the `-nets` parameter.

`first_padring`

Stripes are created within the area enclosed by the pad ring closest to the core area.

`last_padring` Stripes are created within the area enclosed by the pad ring closest to the design boundary.

`design_boundary`

Stripes are created within the design boundary

`-layer` *`layer`*

Specifies the name of the layer in which to create stripes. Stripes can be created on only one layer at a time.

`-master` *`master_cell`*

Specifies the instances of the standard cell (or block) master over which stripes should be generated.

`-max_pin_width` *`value`*

Specifies the maximum pin width in microns. This parameter indicates that only pins less than or equal to this pin width will be considered when generating stripes over pins.

Note: To use this parameter, the `-over_pins` parameter must be set to 1.

Default: `0`, which means that no maximum pin width is set.

`-max_same_layer_jog_length` *`value`*

The maximum length, in micrometers, that a stripe can jog on the same layer before switching to an adjacent layer. For example, if you specify a value of 2 micrometers, and an obstruction causes a stripe on layer M4 to jog for 3 micrometers, the jog will occur on layer M3 or M5. However, if an obstruction causes the stripe to jog for only 1 micrometer, the jog occurs on layer M4.

Default: 2 (If you do not specify this parameter, the stripe automatically switches to an adjacent layer if the jog length is two times the width of the stripe.)

`-max_via_size {`*`shape width% height% target_penetration%`*`}`

The maximum size of a crossover via, specified as a percentage of a full-size via. You must specify three values for each shape that you specify.

The shape values are:

- Pad
- Ring
- Stripe
- Blockring
- Blockpin
- Cover
- No shape

The three values you must specify for each shape are:

- The maximum width for the crossover via, as a percentage of a full-size via
- The maximum height for the crossover via, as a percentage of a full-size via
- The target penetration, which specifies a smaller size for vias connecting to targets of the specified shape

`-merge_stripes_value` *`value`*

Merges a stripe with a nearby block ring if the threshold spacing between the stripe and ring is smaller than the *`value`* (in micrometers) specified with this parameter. The stripe jogs to connect to the nearby ring only on the stripe layer.

Note: To prevent stripes from merging with block rings, set the value of this parameter to `-1`.

Default: 10 (If you do not specify this parameter, stripes do not jog to merge with rings that are more than 10 µm away.)

To merge the stripe on all layers, set the following variable of the setAddStripeOption command:
`-merge_with_all_layers 1`

`-nets {`*`list_of_nets`*`}`

Specifies the names of the nets for which stripes are to be created. The number of net names determines the number of stripes within each set of stripes. The first net in the name list is created first and corresponds to the left or bottom stripe of the set. Separate multiple net names with a space and enclose the list in curly braces, for example, `-nets {net1 net2 net3 net3}`. Braces are not required for one name.

`-number_of_sets` *`integer`*

Specifies the number of stripe sets to be created. You must specify either this parameter or the `-set_to_set_distance` parameter.

Default: If you do not specify this parameter, the number of sets is derived from the `-set_to_set_distance` parameter.

`-orthogonal_only {0 | 1}`

Creates vias *only* for orthogonal wires and pin intersections.

Default: If you do not specify this parameter or if you specify it with a value of `1`, the software creates vias for orthogonal wires and pin intersections.

`-over_bumps {0 | 1}`

Specifies that stripes should be generated over bumps.

Default: `0`

`-over_esd {0 | 1}`

Specifies that stripes should be generated over power/ground pins in an electrostatic discharge cell (ESD).

Default: `0`

`-over_pins {0 | 1}`

Specifies that stripes should be generated over power/ground pins in a standard cell or block.

Default: `0`

`-over_power_domain {0 | 1}`

If set to `1`, adds stripes only within power domain boundaries or the fence around a soft block. For power domains, these stripes start and stop at the power ring that surrounds the selected power domain boundary. For fences, these stripes start and stop at the fence. If both a power domain and a fence are selected, the power domain takes precedence.If you do not specify this parameter or if you specify it with a value of `0`, stripes are prevented from extending to the power domain boundary.

Note: When this option is specified, you can use the `-xleft_offset`, `-xright_offset`, `-ytop_offset`, and `-ybottom_offset` parameters to set boundaries, but you cannot use the `-start_x`, `-start_y`, `-stop_x`, and `-stop_y` parameters to specify an absolute location.

In addition, the software issues a warning message if the specified power and ground nets do not match the power domain's power or ground nets.

Default: `0`

`-padcore_ring_botttom_layer_limit` *`layer_name`*

Specifies the lowest layer that stripes can switch to when encountering a pad or core ring. Stripes that encounter a pad or core ring can only switch to this layer or higher.

`-padcore_ring_top_layer_limit` *`layer_name`*

Specifies the highest layer that stripes can switch to when encountering a pad or core ring. Stripes that encounter a pad or core ring can only switch to this layer or lower.

`-pin_layer` *`layer`*

Specifies the layer on which the stripe is generated.

Note: The layer you specify with this parameter is the same as the layer you specify with the `-layer` parameter.

Default: `TOP`

`-pin_offset`

Offsets the stripe a specified distance from the pin edge to make well tap cell connections. The offset is from the left edge of the stripe to the left edge of the same net pin. This can be a negative number and is in user units.

`-same_layer_target_only {0 | 1}`

Specifies whether to connect to overlapping targets on the same layer only. Specify this parameter with a value of `0` to generate vias that connect to overlapping targets on different layers, even if a target exists on the same layer. Specify this parameter with a value of `1` to connect only to a target on the same layer if such a target exists. If no target exists on the same layer, this parameter does not apply, and it does not make any difference whether the value is set to `0` or `1`.

Default: `0`.

`-same_sized_stack_vias {0 | 1}`

If set to `1`, the software always generates same-sized stacked vias. If set to `0`, the software trims the vias as needed to meet the DRC rules.

Default: `0`

`-set_to_set_distance` *`real`*

Specifies the distance (pitch) from the reference stripe of one set to the reference stripe of the next set. The value of this parameter must be a positive number specified in micrometers. You must specify either this parameter or the `-number_of_sets` parameter.

Default: If you do not specify this parameter, the distance between sets is derived from the `-number_of_sets` parameter.

`-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}`

Prevents vias from being generated on the specified types of pins.

If you specify `-skip_via_on_pin {}`, the power planning software makes connections to all four categories of pins.

Default: If you do not specify this parameter, vias are generated for block pins, pad pins, and cover macro pins, but are not generated for standard cell pins.

`-skip_via_on_wire_shape {[Blockring] [Stripe] [Padring] [Ring] [Noshape]}`

Prevents vias from being generated on the specified wire shapes.

If you specify `-skip_via_on_wire_shape {}`, the power planning software makes connections to all five categories of wire shapes.

Default: If you do not specify this parameter, vias can be generated on all wire shapes *except* no-shape wires.

`-snap_wire_center_to_grid {None | Grid | Half_Grid | Either}`

Controls the snapping of the center of the stripes. If you do not specify this parameter, stripes are not snapped to the routing grid. Specify one of the following:

`None`	Does no snapping of the wires.
`Grid`	Snaps the centers of the wires to the routing grid of the same layer as the wire.
`Half_Grid`	Snaps the centers of the wires to the half-routing grid. The half-routing grid is a virtual grid with the tracks at the center of every two adjacent tracks of the routing grid.
`Either`	Snaps the centers of the wires to the closer of the routing grid or half-grid track.

`-spacing` *`spacing`*

Specifies the edge-to-edge spacing between stripes in each set. You can specify different spacing between each pair of stripes. For *n* number of stripes, you can specify *n* - 1 spacing values. If fewer values are specified, the last-specified spacing is used for the remaining unmatched stripe pairs.

`-split_long_via {`*`threshold_value step_value offset_value`*`}`

Splits vias that are longer than the specified length into smaller vias that have the specified center-to-center step and bottom/left edge offset (all values in micrometers).

Default: If you do not specify this parameter, the software does not split vias.

`-split_vias {0 | 1}`

Specifies whether two or more partial vias can be created at the crossover between a stripe and any target (pin or wire) if an obstruction (macro obstruction; wire or pin on a different net) prevents the creation of a full-size via. This can be useful, for example, if several metal blockages partially obstruct the area where a full-size via would normally be placed. This parameter permits the creation of partial vias that would not violate the process rules, in the open areas. A value of `1` allows multiple partial vias to be created. A value of `0` prevents multiple partial vias from being created.

Default: `0`

`-stacked_via_bottom_layer` *`layername`*

Specifies the lowest layer in which vias can be stacked.

Default: The lowest metal layer in the design.

`-stacked_via_top_layer` *`layername`*

Specifies the highest layer in which vias can be stacked.

Default: The highest metal layer in the design.

`-start_from {bottom | top}`

For horizontal stripes:

- `bottom` indicates that stripes should be generated from bottom to top, taking the bottom offset into account.
- `top` indicates that stripes should be generated from top to bottom, taking the top offset into account.

Default: `bottom`

Note: Offset values you set with this parameter are considered “hard”: The power planning software adheres strictly to the values. However, the opposite values are considered “soft”: Such values could be greater than the hard values, but will never be less than the hard values. For example, if you set a `top` offset value of 4 µm, the `bottom` value will be at least 4 µm, but could be greater.

`-start_from {left | right}`

For vertical stripes:

- `left` indicates that stripes should be generated from left to right, taking the left offset into account.
- `right` indicates that stripes should be generated from right to left, taking the right offset into account.

Default: `left`

Note: Offset values you set with this parameter are considered "hard": The power planning software adheres strictly to the values. However, the opposite values are considered "soft": Such values could be greater than the hard values, but will never be less than the hard values. For example, if you set a `right` offset value of 4 µm, the `left` value will be at least 4 µm, but could be greater.

`-start_x` *`value`* `-start_y` *`value`* `-stop_x` *`value`* `-stop_y` *`value`*

Specifies the exact coordinates that contain the set of stripes to be created in the specified directions. `-start_x` and `-stop_x` correspond to `-direction vertical`. `-start_y` and `-stop_y` correspond to `-direction horizontal`.

Default: If you do not specify this parameter, stripes can be created anywhere in the design.

`-switch_layer_over_obs {0 | 1}`

Controls whether a stripe switches layers, is routed over blocks, and then continues on the original layer.

Using this parameter eliminates the need for a power routing mesh layer, and avoids maximum via stack rule violations.

Default: 0

`-use_interleaving_wire_group {0 | 1}`

If set to `1`, connects multiple wires from the same net together, even if separated by wires from a different net. You can only specify this parameter if you also specify `-use_wire_group 1`. If you do not specify this parameter or if you specify it with a value of `0`, interleaving wire groups are not used.

Default: `0`

`-use_wire_group {0 | 1}`

If set to `1`, connects wires from the same net together. If you do not specify this parameter or if you specify it with a value of `0`, wire groups are not used.

Note: For stripes to connect to rings using wire groups, you must specify the `-useWireGroup` parameter in both the `addRing` and `addStripe` commands.

Default: `0`

`-use_wire_group_bits` *`integer`*

Specifies how many nets are to be included in the wire group. You can only specify this parameter if you also specify `-use_wire_group 1`.

Default: The same number of nets as specified in the `-nets` parameter.

`-via_using_exact_crossover_size {0 | 1}`

Specifies whether to generate partial vias of the exact crossover size. A value of `1` allows partial vias to be generated. A value of `0` prevents partial vias, and the software always generates full-size vias.

Note: This option does not apply when both of the objects at the crossover are rings or stripes, or at the crossover of a ring and stripe. In these cases, the software always generates a full-size via array.

Default: `1`

`-width` *`width`*

Specifies the width of the stripes. You can specify a different width for each net within a set. If the number of widths specified does not match the number of net names, the last-specified width is used for the rest of unmatched nets.

`-xleft_offset` *`value`* `-xright_offset` *`value`*
`-ybottom_offset` *`value`* `-ytop_offset` *`value`*

Defines the start point of the stripes from the core boundary in the specified direction. `-xleft_offset` and `-xright_offset` correspond to `-direction vertical`. `-ybottom_offset` and `-ytop_offset` correspond to `-direction horizontal`.

Note: When used with the `-area` parameter, the offsets are applied relative to the area being striped in the specified direction.

Example

The following command adds stripes in the vertical direction for the `vdd` and `gnd` nets.

```
addStripe -direction vertical -nets {vdd gnd} -width 10 -spacing 1 -layer M6
-xleft_offset 50 -xright_offset 50
```

applyGlobalNets

`applyGlobalNets`

Applies or restores the global net connectivity rules to the design and creates the necessary connections between instances and these global nets.

Parameters

None

checkDrc

```
checkDrc
```

Checks for design rule violations. The software checks for violations of rules governing minimum spacing, minimum width, maximum width, minimum area, and shorts.

Violations are marked in the design display area with design rule checker markers. You can also use the Auto Query feature to read details about the violations. The log file contains a summary message with the number of violations.

Parameters

None

checkDrcInVisibleArea

`checkDrcInVisibleArea`

Checks for design rule violations within the current visible area of the design display area rather than the full design. The software checks for violations of rules governing minimum spacing, minimum width, maximum width, minimum area, and shorts.

Violations are marked in the design display area with design rule checker markers. Click the `Q` button to place the software in *AutoQuery* mode, then hover the cursor over the violation marker. The details about the violation are displayed in the text field to the left of the `Q` button. In addition, the log file contains a summary message with the number of violations.

Parameters

None

clearCutRow

`clearCutRow`

Clears the core rows displayed in the design display area. Use this command after you issue the `cutCoreRow` or `displayCutRow` commands.

Parameters

None

clearDrc

```
clearDrc
```

Clears all design rule checking (DRC) markers in your design.

Note: Do not use the `clearDRC` command before using `ecoRoute` or any other routing command. This is because NanoRoute works only on the existing DRC/violation markers. The routing commands will not necessarily perform DRC checks in the routing halo regions if the violation markers are already cleared.

Parameters

None

clearGlobalNets

`clearGlobalNets`

Resets the global net as well as all of the tie-high and tie-low nets. Use this command after you issue the `globalNetConnections` command.

Parameters

None

connectRingPin

```
connectRingPin
     -nets {list_of_nets}
     {-eachBlock | -selected}
     [-blockSide { left | right | top | bottom}]
     -ringPinTopBottomLayer LayerName
     -ringPinLeftRightLayer LayerName
     -ringPinTopBottomMinWidth real
     -ringPinLeftRightMinWidth real
     -ringPinTopBottomMaxWidth real
     -ringPinLeftRightMaxWidth real
     -ringPinMaxConnections integer
     [-allowRouteOverRows {0 | 1}]
     [-switchLayer {0 | 1}]
     [-stacked_via_top_layer integer]
     [-stacked_via_bottom_layer integer]
     [-via_using_exact_crossover_size {0 | 1}]
     [-same_layer_target_only {0 | 1}]
     [-skip_via_on_wire_shape { [Blockring] [Stripe] [Padring] [Ring] [Noshape] }]
     [-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell] }]
     [-max_via_size {shape width% height% target_penetration%}]
```

Connects ring pins to the first legal target. Legal targets are core rings, block rings, pad rings, stripes, or ring pins that are directly aligned to the sources. Only straight connections are made. If a straight connection is not possible, the pin or area is left open. Use this command after you add power rings and stripes to the design.

Important

The `connectRingPin` command is obsolete. Use the `sroute` command to create power structures.

Parameters

`-allowRouteOverRows {0 | 1}`

Specifies whether to route between blocks if rows are present between the blocks.

Default: If you do not specify this parameter, blocks that have rows between them are not routed to each other.

`-blockSide {[left] [right] [top] [bottom]}`

Specifies which side or sides of the block to connect pins. You can specify one or more sides with this parameter.

Default: If you do not specify this parameter, all pins are connected.

`-eachBlock` Specifies that ring pins for each block are to be connected. You must specify either this parameter or the `-selected` parameter.

`-max_via_size {`*`shape width% height% target_penetration%`*`}`

The maximum size of a crossover via, specified as a percentage of a full-size via. You must specify three values for each shape that you specify.

The shape values are:

- Pad
- Ring
- Stripe
- Blockring
- Blockpin
- Cover
- Noshape

The three values you must specify for each shape are:

- The maximum width for the crossover via, as a percentage of a full-size via
- The maximum height for the crossover via, as a percentage of a full-size via
- The target penetration, which specifies a smaller size for vias connecting to targets of the specified shape.

`-nets {`*`list_of_nets`*`}`

Specifies the net names for the ring pins to be connected. Separate multiple net names with a space and enclose the list in curly braces.

`-ringPinLeftRightLayer` *`LayerName`*

Specifies the layer on which the vertical pins on the left and right sides of the block are to be connected.

Default: If you do not specify this parameter, vertical pins on all layers are connected.

`-ringPinLeftRightMaxWidth` *`real`*

Specifies a maximum width for blockwire connections on the left and right sides of the block that are to be connected. Pins larger than this width are not connected.

Default: If you do not specify this parameter, pins of all widths are connected.

`-ringPinLeftRightMinWidth` *`real`*

Specifies a minimum width for blockwire connections on the left and right sides of the block that are to be connected. Pins smaller than this width are not connected.

Default: If you do not specify this parameter, pins of all widths are connected.

`-ringPinMaxConnections` *`integer`*

Specifies the maximum number of connections to ring pins.

Default: If you do not specify this parameter, all ring pins (except those prevented by another parameter) are connected.

`-ringPinTopBottomLayer` *`LayerName`*

Specifies the layer on which the horizontal pins on the top and bottom sides of the block are to be connected.

Default: If you do not specify this parameter, horizontal pins on all layers are connected.

`-ringPinTopBottomMaxWidth` *`real`*

Specifies a maximum width for blockwire connections on the top and bottom sides of the block that are to be connected. Pins larger than this width are not connected.

Default: If you do not specify this parameter, pins of all widths are connected.

`-ringPinTopBottomMinWidth` *`real`*

Specifies a minimum width for ring pin connections on the top and bottom sides of the block that are to be connected. Pins smaller than this width are not connected.

Default: If you do not specify this parameter, pins of all widths are connected.

`-sameLayerTargetOnly`

Specifies that if a target exists on the same layer, only that same layer target is connected. Vias are not created for targets on different layers.

Default: If you do not specify this parameter and targets exist on both the same layer and different layers, all targets are connected.

`-selected` Specifies that only the pins for blocks selected interactively in the design display area are to be connected. You must specify either this parameter or the `-eachBlock` parameter.

`-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}`

Prevents vias from being generated on the specified types of pins.

If you specify `-skip_via_on_pin {}`, the power planning software makes connections to all four categories of pins.

Default: If you do not specify this parameter, vias are generated for block pins, pad pins, and cover macro pins, but are not generated for standard cell pins.

`-skip_via_on_wire_shape {[Blockring] [Stripe] [Padring] [Ring] [Noshape]}`

Prevents vias from being generated on the specified wire shapes.

If you specify `-skip_via_on_wire_shape {}`, the power planning software makes connections to all five categories of wire shapes.

Default: If you do not specify this parameter, vias can be generated on all wire shapes *except* no-shape wires.

`-stacked_via_bottom_layer` *`layername`*

Specifies the lowest layer in which vias can be stacked.

Default: The lowest metal layer in the design.

`-stacked_via_top_layer` *`layername`*

Specifies the highest layer in which vias can be stacked.

Default: The highest metal layer in the design.

`-switchLayer {0 | 1}`

Specifies whether the wire can switch layers when routing from a ring pin to a target to avoid an obstruction.
Default: `0`

`-via_using_exact_crossover_size {0 | 1}`

Specifies whether to generate partial vias of the exact crossover size. A value of `1` allows partial vias to be generated. A value of `0` prevents partial vias, and the software always generates full-size vias.

Note: This option does not apply when both of the objects at the crossover are rings or stripes, or at the crossover of a ring and stripe. In these cases, the software always generates a full-size via array.

Default: `1`

Example

The following command connects pins for each block.

```
connectRingPin {vdd vss} -eachBlock -ringPinTopBottomLayer M1
-ringpinLeftRightLayer M2
```

connectToGlobalNet

```
connectToGlobalNet
     -connect {Pins | Nets}
     -nets netBasenamePattern
     -pins pinPattern
     -in_instances instanceBasenamePattern
     -tie_highs_lows {0 | 1}
     -to_global_net {powerName | groundName}
     -override_prior_global_connection {0 | 1}
     -under_module hInstanceName
     -verbose {0 | 1}
```

Assigns pins or local power/ground nets to a global power/ground net. You can also use this command to assign tie-high and tie-low pins to global nets.

Note: This command has been replaced by the globalNetConnect command.

Parameters

`-connect {Pins | Nets}`

Specifies whether the pins or nets connect to the global net.
Default: If you do not specify this parameter, pins are connected.

`-in_instances` *`instanceBasenamePattern`*

Specifies the basenames of the instances whose pins are to be connected to the global net. You can use the wildcard (`*`) character to specify a pattern of instance basenames. An instance basename cannot contain the "`/`" character.
Default: If you do not specify this parameter, all instances are connected.

`-nets` *`netBasenamePattern`*

Specifies the basenames of the nets to connect to the global net. You can use the wildcard (`*`) character to specify a pattern of net basenames. An net basename cannot contain the "`/`" character.
.*Default:* None

Note: If you do not specify this parameter, a warning message is issued.

`-override_prior_global_connection {0 | 1}`

Specifies whether to disconnect pins or local nets from the old global nets, and then reconnect them to the global net specified in this command. If set to 0, a pin or a local net already connected to a global net is left alone.
Default: `0`

`-pins` *`pinPattern`*
Specifies the names of the pins to connect to the global net. You can use the wildcard (`*`) character to specify a pattern of pin names.
Default: None

Note: If you do not specify this parameter, a warning message is issued.

`-tie_highs_lows {0 | 1}`

Specifies whether the tie-high pins and tie-low pins are connected to the global net specified in the `-to_global_net` parameter.
Default: If you do not specify this parameter, the software uses a value of `0` and the tie-high and tie-low pins are not connected.

`-to_global_net {`*`powerName`* `|` *`groundName`*`}`

Specifies the name of power or ground net.

`-under_module` *`hInstanceName`*

Specifies the root of the hierarchy inside which the pins or local nets are to be connected to the specified global net.
Default: The top module of the design

`-verbose {0 | 1}`
Specifies whether to display more-detailed messages on the console. This parameter can be useful during troubleshooting.
Default: If you do not specify this parameter, the software uses a value of `0`.

Example

The following command connects the specified component and pins to the global net:

```
connectToGlobalNet -tie_highs_lows 0 -to_global_net VDD -verbose 0
-override_prior_global_connection 0 -in_instances * -nets {} -pins VDD
-connect Pins -under_module {}
```

In this example, `connectToGlobalNet` connects all instances (`*`) in all modules with pins `VDD` to the previously-defined global net `VDD`.

cutCoreRow

```
cutCoreRow
     [-noPlacementObs]
     [-cutBumpObs siteName]
```

Note: The `cutCoreRow` command is obsolete and will be removed in the next major release of the software. To cut site rows that intersect with a specified area or object, use the `cutRow` command instead.

Cuts the rows based on the placement blockages or block halos in the floorplan, or based on bumps. After the cut is completed, the updated rows are highlighted in the design display area. Use this command whenever you make a change to the floorplan, such as moving a hard block or creating placement obstructions.

Parameters

`-cutBumpsObs` *`siteName`*

Specifies the site name of the rows for which the space occupied by bumps will be cut.

Use this parameter to prevent registers (that are associated with *`siteName`*) from being placed under bumps.

Note: The size of the site needs to be at least as large as the register—otherwise, registers will protrude into a bump.

`-noPlacementObs`

Specifies that cut rows are to be based only on block halos. Placement blockages are ignored.

Default: This parameter is turned off.

Example

The following command cuts a space for the special site `FF_CORE`:

```
cutCoreRow -cutBumpObs FF_CORE
```

displayCutRow

`displayCutRow`

Displays the core rows that were cut when you issued the `cutCoreRow` command.

Parameters

None

editPowerVia

```
editPowerVia
     {-add_vias {0 | 1}
          | -modify_vias {0 | 1}
          | -delete_vias {0 | 1}}
          | -convert_to_real_vias {0 | 1}
          | -fix_mincut_vias {0 | 1}
     [{-entire_design
          | -between_selected_wires {0 | 1}
          | -selected_blocks {0 | 1}
          | -area x1 y1 x2 y2}]
     [{-via_scale_height integer -via_scale_width integer
          | -via_rows integer -via_columns integer
          | -via_height real_number -via_width real_number}]
     [-top_layer layername]
     [-bottom_layer layername]
     [-via_using_exact_crossover_size {0 | 1}
     [-orthogonal_only [0 | 1]
     [-split_long_via {value_1 value_2}
     [-split_vias {0 | 1}
     [-same_sized_stack_vias {0 | 1}]
     [-skip_via_on_wire_shape {[Blockring] [Stripe] [Followpin]
          [Corewire] [Blockwire] [Iowire] [Padring] [Ring] [Fillwire] [Noshape]}
     [-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}]
```

Adds power vias to the design or performs one of the following actions to existing power vias:

- Modifies
- Deletes
- Converts virtual vias created by the Virtual Power Planning software (VPP) to real vias
- Fixes vias that violate the LEF `MINIMUMCUT` rule

By default, the action you specify applies to all vias in the design.

Parameters

`-add_vias {0 | 1}`	Adds power vias in your design. *Default:* `0`
`-area x1 y1 x2 y2`	Adds, modifies, or deletes power vias within the coordinates that you specify.

`-between_selected_wires {0 | 1}`

Adds, modifies, or deletes power vias between selected wires or pins in your design. You must select these interactively in the design display area.

Default: `0`

`-bottom_layer` *`layername`*

Specifies the lowest layer to which vias can connect.
Default: The lowest metal layer in the design.

`-convert_to_real_via {0 | 1}`

Converts virtual vias created by the Virtual Power Planning (VPP) software to real vias. This parameter must be used in conjunction with either the `-area` or the `-entire_design` parameter.

Default: `0`

`-delete_vias {0 | 1}`

Deletes power vias from your design.

Default: `0`

`-entire_design {0 | 1}`

Performs the action that you specify to the entire design area.

Default: 1

`-fix_mincut_vias {0 | 1}`

Detects vias that violate the LEF `MINIMUMCUT` rule and fixes these violations. You must run the Verify Geometry command before you issue the editPowerVias command with this parameter. This parameter must be used in conjunction with either the `-area` or the `-entire_design` parameter.

Default: `0`

`-orthogonal_only [0 | 1]`

Creates vias *only* for orthogonal wires and pin intersections.

Default: If you do not specify this parameter or if you specify it with a value of `1`, the software creates vias for orthogonal wires and pin intersections.

`-same_sized_stack_vias {0 | 1}`

If set to `1`, the software always generates same-sized stacked vias. If set to `0`, the software trims the vias as needed to meet the DRC rules.

Default: `0`

`-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}`

Prevents vias from being generated on the specified types of pins.

If you specify `-skip_via_on_pin {}`, the power planning software makes connections to all four categories of pins.

Default: If you do not specify this parameter, vias are generated for block pins, pad pins, and cover macro pins, but are not generated for standard cell pins.

`-skip_via_on_wire_shape {[Blockring] [Stripe] [Followpin] [Corewire] [Blockwire] [Iowire] [Padring] [Ring] [Fillwire] [Noshape]}`

Prevents vias from being generated for the specified wire shapes.

Default: If you do not specify this parameter, vias can be generated for all wire shapes.

`-split_long_via {`*`value_1 value_2`*`}`

Splits vias that are longer than the specified length (first value, in micrometers) into smaller vias that have the specified edge-to-edge spacing (second value, in micrometers).

Default: If you do not specify this parameter, the software does not split vias.

`-split_vias {0 | 1}`

Specifies whether two or more partial vias can be created at the crossover between a stripe and any target (pin or wire) if an obstruction (macro obstruction; wire or pin on a different net) prevents the creation of a full-size via. This can be useful, for example, if several metal blockages partially obstruct the area where a full-size via would normally be placed. This parameter permits the creation of partial vias that would not violate the process rules, in the open areas. A value of `1` allows multiple partial vias to be created. A value of `0` prevents multiple partial vias from being created.

Default: `0`

`-top_layer` *`layername`*

Specifies the highest layer to which vias can connect.

Default: The highest metal layer in the design.

`-via_height` *`real_number`*
`-via_width` *`real_number`*

Specifies the absolute height and width of the added or modified power vias. You can use these parameters only if you choose the `-add_vias` or `-modify_vias` parameter.
Default: `0.0`

`-via_rows` *`integer`*
`-via_columns` *`integer`*

Specifies the number of cuts to make in the power vias. You can use these parameters only if you choose the `-add_vias` or `-modify_vias` parameter.
Default: `0`

`-via_scale_height` *`integer`*
`-via_scale_width` *`integer`*

Specifies the percentage to scale the power via dimensions. You can use these parameters only if you choose the `-modify_vias` or `-add_vias` parameter.

`-via_using_exact_crossover_size {0| 1}`

Specifies whether to generate partial vias of the exact crossover size. A value of `1` allows partial vias to be generated. A value of `0` prevents partial vias, and the software always generates full-size vias.

Note: This option does not apply when both of the objects at the crossover are rings or stripes, or at the crossover of a ring and stripe. In these cases, the software always generates a full-size via array.

Default: `1`

Example

The following command creates a power via between selected wires with a cut pattern of 3 by 3:

```
editPowerVia -add_vias 1 -between_selected_wires 1 -via_rows 3 -via_columns 3
```

globalNetConnect

```
globalNetConnect
     globalNetName
     {{-type pgpin -pin pinNamePattern |
     -type tiehi [-pin pinNamePattern] |
     -type tielo [-pin pinNamePattern]}
     {{-singleInstance | -singleInst | -sinst} instName |
     [{-instanceBasename | -instBasename | -inst} instBasenamePattern]
     [{-hierarchicalInstance | -hierInst | -module} hierInstName |
     -region llx lly urx ury |
     -powerDomain powerDomainName | -all]} |
     -type net -net netBasenamePattern
     [{-hierarchicalInstance | -hierInst | -module} hierInstName |
     -powerDomain powerDomainName | -all]}
     [-override]
     [-verbose]
```

Adds a new global net connection to the specified global net.

Note: This command replaces the connectToGlobalNet command.

You can use the `globalNetCommand` syntax in three general design situations. The three design situations, and the suggested usage for the `globalNetCommand` syntax, appear below:

- Connecting pins in a single instance to a global net. In the example below, *instName* represents an instance's complete hierarchical name, without any wildcard characters:

  ```
  globalNetConnect
  globalNetName
  {-type pgpin -pin pinNamePattern | -type tiehi [-pin pinNamePattern]
  | -type tielo [-pin pinNamePattern]}
  -singleInstance instName
  [-override]
  [-verbose]
  ```

- Connecting pins in multiple instances, or pins in a region, to a global net:

  ```
  globalNetConnect
  globalNetName
  {-type pgpin -pin pinNamePattern | -type tiehi [-pin pinNamePattern]
  | -type tielo [-pin pinNamePattern]}
  [-instanceBasename instBasenamePattern]
  [-hierarchicalInstance hierInstName | -region llx lly urx ury | -all]
  [-override]
  [-verbose]
  ```

- Connecting nets to a global net:

  ```
  globalNetConnect
  globalNetName
  -type net -net netBasenamePattern
  [-hierarchicalInstance hierInstName | -all]
  ```

```
   [-override]
   [-verbose]
```

Parameters

`-all` — Applies the global net connection to all instances in the design.

`globalNetName` — The name of the global net to which the specified pins, modules, or nets connect.

`-hierarchicalInstance` *`hierInstName`*

Applies the global net connection to instances under *`hierInstName`*. Only the pins of the instances or the local nets under the specified module or hierarchical instance are connected.

This parameter is mutually exclusive with the `-region` parameter.

Note: You can also use the following spelling variations for this argument:

- `-hierInst`
- `-module`

`-instanceBasename` *`instBasenamePattern`*

Specifies the names of leaf instances for which pins are to be connected to the global net. You can use the wildcard (*) character to specify a pattern of instance basenames. An instance basename cannot contain the "/" character. This parameter is used in conjunction with the `-type pgpin` parameter, or with the `-type tiehi` or `-type tielo` parameters.

Note: You can also use the following spelling variations for this argument:

- `-inst`
- `-instBasename`

`-net` *`netBaseName`*

Specifies the basenames of nets to connect to the global net. You can use the wildcard (`*`) character to specify a pattern of net basenames. An net basename cannot contain the "`/`" character. This parameter is used in conjunction with the `-type net` parameter.

`-override`

Specifies that the values that used with the `globalNetConnect` command override global net connection values that have been set previously.

`-pin` *`pinNamePattern`*

Specifies the pins to connect to the global net. You can use the wildcard (`*`) character to specify a pattern of pin names. This parameter is used in conjunction with the `-type pgpin` parameter, or optionally with the `-type tiehi` or `-type tielo` parameters.

`-powerDomain` *`powerDomainName`*

Applies the global net connection to instances within the specified power domain.

`-region` *`llx lly urx ury`*

Applies the global net connection to instances inside the specified region. The region is specified by the `x` and `y` coordinates of the lower left and upper right boundary. This parameter is mutually exclusive with the `-module` parameter.

`-singleInstance` *`instName`*

Connects pins in a single instance to a global net. *`-instName`* is the instance's complete hierarchical name, without any wildcard characters.

Note: You can also use the following spelling variations for this argument:

- `-sinst`
- `-singleInst`

`-type {net | pgpin | tielo | tiehi}`

Specifies the type of connection to be made to the global net. Use one of the following values:

	`net`	Specifies that the nets listed with the `-net` parameter are to be connected.
	`pgpin`	Specifies that the power and ground pins listed with the `-pin` parameter are to be connected.
	`tielo`	Specifies that tie low pins be connected.
	`tiehi`	Specifies that tie high pins be connected.
`-verbose`		Specifies that connection statistics and warning messages are displayed in the console. This parameter can be useful during troubleshooting.

Example

The following command connects the vdd! net with the VDD global net:

```
globalNetConnect VDD -type net -net vdd!
```

limitPowerPlannerMessage

```
limitPowerPlannerMessage
     messageID
     messageLimitCount
```

Limits the number of times the tool can issue a specified error message.

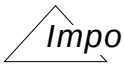
Parameters

messageID	Specifies the ID of the message you want to limit.
messageLimitCount	Specifies the number of times the tool can issue the message.

loadEMLimits

`loadEMLimits` *`fileName`*

Loads the electromigration limits that were saved to a `.dsf` file with the *Specify Electromigration Limit* form (*Power - Power Planning - Specify Electromigration*) in the GUI.



Important

You should not modify the `.dsf` file manually as this may cause data integrity issues within Encounter.

Parameters

`fileName`	Specifies the path to the `.dsf` file that contains the electromigration limits.

loadSpecialRoute

`loadSpecialRoute` *`filename`*

Clears all special routes and runs the `addSpecialRoute` command. If the specified filename is compressed (with the `.gz` extension), the file is read in directly.

Parameters

`filename`	Specifies the name of the file that contains special route information, to be used as input for the design.

optimizePowerPlan

```
optimizePowerPlan
     {-method totalpower -totalPower milliwatts
     | -method estimate -estimatePowerFile estimateFileName
     | -method dummy_clock -clockRate megahertz -toggleProbability value
     | -method railonly -readInstancePower instancePowerFileName
     | -method post_cts_clock -toggleFile toggleFileName}
     -padFile padFileName1 padFileName2 …
     -nets netNames
     -mode floorplan | layout
     [-irDropThreshold volts]
     [-layers layerNames]
     [-shapes shapeNames]
     [-template templateName]
     [-templateParameterFile templateParameterFileName]
```

Important

The `optimizePowerPlan` command is obsolete. Use the `addStripe` and `addRing` commands to create power structures.command instead.

Uses the wire editor or automatic power planning (APP) to modify the wire width of special routes to eliminate IR drop errors in a design. Use this command after running the synthesizePowerPlan command.

The `optimizePowerPlan` command can be run in one of five optimization modes based on—

- Total power
- An estimate power parameter file
- A dummy clock
- An instance power file

Important

The `optimizePowerPlan` command changes the wire width of the layout. After wire widths have been changed, you cannot undo the change. Cadence strongly recommends that you save power structures before using the `optimizePowerPlan` command.

Note: If you want the power planning software to consider electromigration during optimization, you must use the GUI to optimize your power plan. Specifically, you must use the *Perform EM optimization* option on the *Advanced* tab of the *Optimize Power Plan* form.

The power planning software behaves differently when you specify the `-layers` and `-shapes` parameters together, individually, or not at all:

- `-layers` specified alone: `-layers M4 M5`

  The software scales the width of any of the four power routing shapes (`CORERING`, `BLOCKRING`, `HORIZONTALSTRIPE`, `VERTICALSTRIPE`) on layers `M4` and `M5` only.

- `-shapes` specified alone: `-shapes`

  The software scales the width of any of the four power routing shapes on any layer containing them.

- `-layers` and `-shapes` specified together: `-layers M4 M5 -shapes CORERING`

  The software scales the width of core rings on layers `M4` and `M5`.

- Neither `-layers` nor `-shapes` specified.

  The software scales the width of any of the four power routing shapes on any layers that contain them.

Parameters

`-irDropThreshold` *`volts`*

Specifies the maximum IR drop value allowed for the design (in volts).

If the original layout's IR drop is less than the IR drop threshold you specify, the power planning software scales the wire segment down until the analyzed IR drop falls below the specified threshold.

If the original layout's IR drop is greater than the IR drop threshold you specify, the power planning software scales the wire segment up until the analyzed IR drop exceeds the specified threshold.

Whether wire width is scaled up or down, the IR drop associated with any optimized wire segment will not be higher than the IR drop threshold.

`-layers` *`layerNames`* Optimizes only the specified layers.

Specify one or more layers using generic layer names `M1`, `M2`, and so on.

`-method dummy_clock -clockRate` *`megahertz`* `-toggleProbability` *`value`*

Performs power analysis on all nets, including clock nets, toggling at a specified clock rate and a specified toggle probability for all nets.

When you use this method, you must also specify the following parameters:

`-clockRate` *`megahertz`*

The rate at which the net toggles, in megahertz.

Default: `100`

`-toggleProbability` *`value`*

Defines the percentage of nets toggling in the entire design, including all the clock nets.

Default: `0.2`.

`-method estimate -estimatePowerFile` *`estimateFileName`*

Specifies the name of a file that contains parameters used by the power planning software to estimate power consumption. This provides a more accurate estimate than using `-method totalpower`.

`-method post_cts_clock -toggleFile` *`toggleFileName`*

Performs power analysis with clock nets (based on a net toggle probability file) at a set clock rate (in megahertz) for each clock domain, and a set toggle probability (in percent) for the clock's data paths.

`-method railonly -readInstancePower` *`instancePowerFileName`*

Specifies the output instance power file (*`filename`*`.power`).

The instance power file has the following format:

```
clock_rate megahertz
toggle_probability value
load_capacitance picofarads
multiplier value
slew nanoseconds
```

For example:

```
clock_rate 100
toggle_probability 1.0
load_capacitance 0.1
multiplier 0.2
slew 0.120
```

`-method totalpower -totalPower` *`milliwatts`*

Specifies the total average power for the chip or the block.

If you specify a value, the power planning software runs area-based rail analysis. You do not need an instance power file to run in this mode.

`-mode floorplan | layout`

Runs power analysis either early in the floorplanning cycle (`floorplan`), or with power structures completely designed (`layout`).

`-nets` *`netNames`* Specifies the power net name or list of names to be analyzed.

`-padFile` *`padFileName1 padFileName2 …`*

Specifies one or more pad location files.

You must specify a separate pad location file for each net in the design.

`-shapes` *`shapeNames`*

Optimizes only the specified power routing shapes.

Specify one or more of the following power routing shapes:

- `CORERING`
- `BLOCKRING`
- `VERTICALSTRIPE`
- `HORIZONTALSTRIPE`

`-template` *`templateName`*

Specifies the template file (`.dpt`) created during the synthesize power plan process.

`-templateParameterFile` *`templateParameterFileName`*

Specifies the template parameter file (`.tParam`) created during the synthesize power plan process.

Example

The following command illustrates using the `optimizePowerPlan` command in `totalpower` mode:

```
optimizePowerPlan -totalPower 200 -method totalpower \
    -padFiles {oppAutoGen.VSS.pp oppAutoGen.VDD.pp} \
    -mode floorplan \
    -nets {VSS VDD} \
    -irDropThreshold .05 \
    -template dt2 -templateParameterFile dtmf_chip.dt2.tParam
```

repairPowerWire

```
repairPowerWire
     [-blocks name | -selected {0 | 1} | -area {x1 y1 x2 y2}]
     [-allow_route_over_rows {0 | 1}]
     [-route_over_rows_only {0 | 1}]
     [-break_stripes_at_block_rings {0 | 1}]
     [-break_if_overlap_ringpin {0 | 1}]
     [-jog_to_connect {0 | 1}]
     [-merge_stripes_value real_number]
     [-stacked_via_top_layer layername]
     [-stacked_via_bottom_layer layername]
     [-via_using_exact_crossover_size {0 | 1}]
     [-split_vias {0 | 1}
     [-same_sized_stack_vias {0 | 1}]
     [-same_layer_target_only {0 | 1}]
     [-skip_via_on_wire_shape { [Blockring] [Stripe] [Padring] [Ring] [Noshape]}]
     [-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}]
     [-max_via_size {shape width% height% target_penetration%}]
```

Regenerates power stripes at the same locations in your design based on changes in the floorplan such as block movements. Whenever you move a hard block, blockages and pins inside the block might violate design rules governing spacing or shorts. Use this command after creating stripes with the `addStripe` command.

Parameters

`-allow_route_over_rows {0 | 1}`

If set to `1`, creates stripes over the rows. If set to `0`, prevents stripes from being created over rows, and stops stripes at the last legal target, which is either a block ring, core ring, or pad ring.

Default: `1`

`-area {x1 y1 x2 y2}`

Repairs violations within the specified coordinates (*x1 y1 x2 y2*). The coordinate values must be enclosed within curly braces.

`-blocks name`

Repairs violations on stripes crossing the named block in your design.

`-break_if_overlap_ringpin {0 | 1}`

Specifies whether to prevent stripes from overlapping with same direction ring pins of a different net. If you do not specify this parameter, or if you specify it with a value of `0`, stripes can overlap with same direction ring pins of a different net.

Default: `1`

`-break_stripes_at_block_rings {0 | 1}`

Specifies whether to break power stripes at block rings based upon obstruction layers inside the block.

Default: `0`

`-jog_to_connect {0 | 1}`

Specifies whether to permit stripes to jog in order to connect with a block ring. A value of `1` allows stripes to jog. A value of `0` prevents stripes from jogging.

Default: `0`

`-max_via_size {`*`shape width% height% target_penetration%`*`}`

The maximum size of a crossover via, specified as a percentage of a full-size via. You must specify three values for each shape that you specify.

The shape values are:

- Pad
- Ring
- Stripe
- Blockring
- Blockpin
- Cover
- Noshape

The three values you must specify for each shape are:

- The maximum width for the crossover via, as a percentage of a full-size via
- The maximum height for the crossover via, as a percentage of a full-size via
- The target penetration, which specifies a smaller size for vias connecting to targets of the specified shape.

`-merge_stripes_value` *`real_number`*

Specifies a distance for merging power stripes with nearby rings. Stripes will jog slightly for ring connections based upon the specified merge threshold to merge into the ring corners.

`-route_over_rows_only {0 | 1}`

If set to `1`, regenerates stripes only over rows. If set to `0`, stripes are not limited to being regenerated over rows only.

Default: `0`

`-same_layer_target_only {0 | 1}`

Specifies whether to connect to overlapping targets on the same layer only. Specify this parameter with a value of `0` to generate vias that connect to overlapping targets on different layers, even if a target exists on the same layer. Specify this parameter with a value of `1` to connect only to a target on the same layer if such a target exists. If no target exists on the same layer, this parameter does not apply, and it does not make any difference whether the value is set to `0` or `1`.

Default: `0`

`-same_sized_stack_vias {0 | 1}`

If set to `1`, the software always generates same-sized stacked vias. If set to `0`, the software trims the vias as needed to meet the DRC rules.

Default: `0`

`-selected {0 | 1}` Specifies whether to repairs violations on stripes crossing the block that you selected in the GUI.

Default: `0`

`-skip_via_on_pin {[Pad] [Block] [Cover] [Standardcell]}`

Prevents vias from being generated on the specified types of pins.

If you specify `-skip_via_on_pin {}`, the power planning software makes connections to all four categories of pins.

Default: If you do not specify this parameter, vias are generated for block pins, pad pins, and cover macro pins, but are not generated for standard cell pins.

`-skip_via_on_wire_shape {[Blockring] [Stripe] [Padring] [Ring] [Noshape]}`

Prevents vias from being generated on the specified wire shapes.

If you specify `-skip_via_on_wire_shape {}`, the power planning software makes connections to all five categories of wire shapes.

Default: If you do not specify this parameter, vias can be generated on all wire shapes *except* no-shape wires.

`-split_vias {0 | 1}`

Specifies whether two or more partial vias can be created at the crossover between a stripe and any target (pin or wire) if an obstruction (macro obstruction; wire or pin on a different net) prevents the creation of a full-size via. This can be useful, for example, if several metal blockages partially obstruct the area where a full-size via would normally be placed. This parameter permits the creation of partial vias that would not violate the process rules, in the open areas. A value of `1` allows multiple partial vias to be created. A value of `0` prevents multiple partial vias from being created.

Default: `0`

`-stacked_via_bottom_layer` *`layername`*

Specifies the lowest layer in which vias can be stacked.

Default: The lowest metal layer in the design.

`-stacked_via_top_layer` *`layername`*

Specifies the highest layer in which vias can be stacked.

Default: The highest metal layer in the design.

`-via_using_exact_crossover_size {0 | 1}`

Specifies whether to generate partial vias of the exact crossover size. A value of `1` allows partial vias to be generated. A value of `0` prevents partial vias, and the software always generates full-size vias.

Note: This option does not apply when both of the objects at the crossover are rings or stripes, or at the crossover of a ring and stripe. In these cases, the software always generates a full-size via array.

Default: `1`

saveSpecialRoute

`saveSpecialRoute` *`filename`*

Saves special route information in a file. If the specified filename is compressed (with the `.gz` extension), the file is gzipped.

Parameters

`filename`	Specifies the name of the output file into which special route information is saved. The file is saved in the `.spr` format.

selectRow

```
selectRow
     x1 y1
     {1 | 0}
```

Selects the row at the coordinates you specify, enabling you to issue an additional command for the row. Use this command before you issue the `addRing` command if you want a rectilinear ring around a row cluster.

Parameters

x1	Specifies the x coordinate, in micrometers, for the row to be selected.
y1	Specifies the y coordinate, in micrometers, for the row to be selected.
{1 | 0}	If 1 is specified, the newly selected row is appended to the selected set or the selected set is refreshed, starting with the new row. If 0 is specified, the selected set is cleared and the new row is added.

Example

The following command appends a row to the selected set:

```
selectRow 7216.926 6423.550 1
```

setAddRingOption

```
setAddRingOption
     [-reset_default]
     [-avoid_short {0 | 1}
     [-extend_blockring_search_distance distance]
     [-extend_corering_search_distance float]
     [-extend_merge_with_prewires {0 | 1}]
     [-extend_over_row {0 | 1}]
     [-extend_stripe_search_distance float]
     [-extend_search_nets {* | {net_names}}]
     [-ignore_rows {0 | 1}
     [-ring_target {stripe | pad_ring | core_ring | first_ring | default}]
     [-skip_crossing_trunks {horizontal | vertical}]
```

Sets global variables for block rings and core rings when you use the addRing command.

Parameters

`-avoid_short {0 | 1}`

Specifies whether to prevent a ring or a ring segment from being created if it would cause a DRC violation to occur. Specify `1` to prevent the creation of rings that would cause a DRC violation. Specify `0` to enable the creation of rings that cause a DRC violation.
Default: `0`

`-extend_blockring_search_distance` *`distance`*

Extends the block ring search a specified distance beyond the initial block ring target to enable the connection of multiple block ring targets. For example, if VDD consists of three bits (segments) running parallel to one another, each having a width of two microns and a spacing between them of one micron, using this variable with a specified distance of eight microns will extend the block ring search a distance of eight microns to enable a connection to the multiple block ring targets. The *`distance`* is specified as the measured distance from the first to last bits. Units in microns.

Note: You must specify this variable before using the `addRing` command. When using the `addRing` command, you must include the following parameters:

```
-use_wire_group_bits 1
-use_wire_group 1
```

`-extend_corering_search_distance` *`float`*

Specifies the wire group search distance for connecting all bits of the wire group stripes.

Default: `0.0`

`-extend_merge_with_prewires {0 | 1}`

Automatically merges newly-created ring wires with samenet prerouted wires.

Default: `1`

`-extend_over_row {1 | 0}`

If 1 is specified, the cut row is not considered a blockage by the addRing command, which generates the ring over the cut row. If `0` is specified, the cut row is considered a blockage and the ring is not generated over the row. Rings are created based on the parameters that you specify in the addRings command.
Default: `0`

`-extend_search_nets {* | {net_names}}`

Specifies the net names to append to the wire group search distance properties.

`-extend_stripe_search_distance` *`float`*

Specifies the wire group search distance for connecting all bits of the wire group core rings.

Default: `0.0`

`-ignore_rows {0 | 1}`

Specifies whether the power planning software ignores all rows (cut rows and native rows) when rings are created. Specify `1` to create rings that ignore rows. Specify `0` to create rings that do not ignore rows.
Default: `0`

`-reset_default`

Resets all parameters to their default values.

`-ring_target {stripe | pad_ring | default}`

`stripe`	Both rings and stripes are considered targets.
`pad_ring`	Ring wires can be extended to pad rings.
`core_ring`	Ring wires can consider core rings as targets, even when core rings are inside the I/O pad area.
`first_ring`	Ring wires consider either the first core ring or the pad ring as a target.
`default`	Only rings are considered targets.

Note: You can specify that rings are extended to both pad rings and to stripes. To do this you must specify the ring_target parameter twice, as show in the following example:

```
-ring_target stripe
-ring_target pad_ring
```

`-skip_crossing_trunks {horizontal | vertical | none}`

Removes the horizontal or vertical portion of rings (shared or inside edges) when used in conjunction with the `addRing -block_rings -around shared_cluster` command.

Example

The following command specifies that rings can use stripes as targets and that the rings generated around clusters of block rings with shared edges do not have horizontal wires between the blocks.

```
setAddRingOption -ring_target stripe -skip_crossing_trunks horizontal
```

setAddStripeOption

```
setAddStripeOption
     [-reset_default]
     [-allow_nonpreferred_dir {0 | 1}]
     [-break_at_outer_ring {0 | 1}]
     [-break_if_overlap_ringpin {0 | 1}]
     [-break_outside_ringmacro {0 | 1}]
     [-domain_offset_from_core {0 | 1}]
     [-extend_to_closest_target {none | stripe | ring}
     [-extend_to_first_ring {0 | 1}]
     [-ignore_block_check {0 | 1}]
     [-ignore_nondefault_domains {0 | 1}]
     [-ignore_blocking_when_breaking {0 | 1}]
     [-merge_with_all_layers {0 | 1}]
     [-offset_from_core {0 | 1}]
     [-over_row_extension {0 | 1}]
     [-remove_floating_stripe_over_block {0 | 1}]
     [-route_over_rows_only {0 | 1}]
     [-rows_without_stripes_only {0 | 1}]
     [-spacing_from_block float]
     [-split_wire_spacing floating_point_value]
     [-split_wire_weight integer_0_to_10]
     [-split_wire_width floating_point_value]
     [-stop_at_last_wire_for_area {0 | 1}]
     [-stripe_min_length float]
     [-switch_cellname {cellA cellB ...}]
     [-switch_layer_overlap_length floating_point_value]
     [-trim_antenna_back_to_shape {block_ring | core_ring | pad_ring | stripe
          | standard_cell_pin}]
     [-trim_antenna_max_distance distance]
     [-use_exact_spacing {0 | 1}]
     [-use_point2point_router {0 | 1}]
     [-use_stripe_width {0 | 1}]
```

Sets global variables for power stripes when you use the addStripe command.

Parameters

`-allow_nonpreferred_dir {0 | 1}`

If set to `1`, the power stripe ignores the preferred direction of the layer when switching to different layers. If set to `0`, the power stripe uses the preferred direction of the layer when switching to different layers

Default: `0`

`-break_at_outer_ring {0 | 1}`

If set to `1`, when a stripe encounters nested block rings or ring pins that are not in the same wire group, the stripe breaks at the outer ring. If set to `0`, the stripe breaks at the innermost ring.

Note: This parameter only works in conjunction with the following parameters of the `addStripe command:`

- `-break_stripes_at_block_rings 1`
- `-break_at_selected_blocks 1`

Default: `0`

`-break_if_overlap_ringpin {0 | 1}`

If set to `0`, prevents stripes from overlapping with same direction ring pins of a different net. If set to `1`, stripes can overlap with same direction ring pins of a different net.

Note: This option is only valid if the ring pin is more than half the length of the macro in the direction of the ring pin.

Default: `0`

`-break_outside_ringmacro {0 | 1}`

If set to `1`, the stripe breaks a small distance outside the block if the selected block has ring pins. If set to `0`, the stripe connects to the ring pins. This parameter is available only if you also specify `addStripe -break_at_selected_blocks`.

`-domain_offset_from_core {0 | 1}`

Creates stripes over a power domain to align with the global stripes in your design by enabling you to use the offset that was used for the global stripes as the offset for the power domain stripes. This variable uses the offset from the core, which is the default for global stripes.

Default: `0`

`-extend_to_closest_target {none | stripe | ring}`

Extends the stripe to the specified target.

Default: none

`-extend_to_first_ring {0 | 1}`

If set to `1`, stripe antennas are extended to the closest ring. If set to `0`, stripes stop at the boundary specified by the `-extend_to` parameter of the addStripe command.

Default: `0`

`-ignore_block_check {0 | 1}`

If set to `1`, when a stripe encounters a ring, the software does not check whether that ring is surrounding a block. If set to `0`, when a stripe encounters a ring, the software automatically checks whether a block is enclosed within the ring.

Note: This parameter only works in conjunction with the following parameters of the `addStripe command:`

- `-break_stripes_at_block_rings 1`
- `-break_at_selected_blocks 1`

Default: `0`

`-ignore_blockring_when_breaking`

If set to `1`, breaks the stripe outside the block if the stripe cannot go over the block.

If set to `0`, breaks the stripe at the block ring if it cannot go over the block.

Default: `0`

`-ignore_nondefault_domains {0 | 1}`

Creates global stripes over domains without breaking. If you do not want the stripes to break at the non-default domains, such as when the specified nets are not all included in the non-default domain, set this variable to `1` before issuing the `addStripe` command.

Default: `0`

`-max_same_layer_jog_length` *`distance`*

Specifies the maximum length, in micrometers, that a stripe can jog on the same layer before switching to an adjacent layer. For example, if you specify a value of 2 micrometers, and an obstruction causes a stripe on layer `M4` to jog for 3 micrometers, the jog will occur on layer `M3` or `M5`. However, if an obstruction causes the stripe to jog for only `1` micrometer, the jog occurs on layer `M4`.

Default: The stripe automatically switches to an adjacent layer if the jog length is two times the width of the stripe.

`-merge_with_all_layers {0 | 1}`

Merges a stripe to a block ring or pin on all layers. This parameter is similar to the `-merge_stripes_value` parameter of the `addStripe` command, except that it merges a stripe to all layers instead of only the stripe layer.

Default: `0`

`-offset_from_core {0 | 1}`

If set to `1`, the first or last stripe is offset from the core area of the design. If set to `0`, the first or last stripe is offset from core ring, pad ring, or design boundary, depending on the parameters specified with the addStripe command.

Note: This parameter does not affect the offset if you specify the addStripe command with any of the following parameters:

- `-over_power_domain 1`
- `-extend_to all_domains`
- `-area`

Default: `1`

`-over_row_extension {0 | 1}`

Extends the stripes to cover the followpin at the row end.

Default: `0`

`-remove_floating_stripe_over_block {0 | 1}`

If set to `1`, stripe fragments that start and end inside the same block or ring macro are removed. If set to `0`, stripe fragments are not removed.

Default: `1`

`-reset_default`

Resets all parameters to their default values.

`-route_over_rows_only {0 | 1}`

If set to `1`, routes stripes only over rows within the boundary established with the addStripe command. If set to `0`, stripes are not limited to being routed over rows within this boundary.

Default: `0`

`-rows_without_stripes_only {0 | 1}`

If set to `1`, global stripes are generated as specified by other parameters. Then stripes are also generated over row clusters that do not already contain stripes. If set to `0`, stripes are generated as specified by other parameters and not over row clusters without stripes.

Default: `0`

`-spacing_from_block` *`float`*

If the value is less than 0, over-the-block stripes end the specified distance from the boundary.

If the value is greater than 0, global stripes break at the specified user units from the obstructed blocks.

If the value is equal to 0, the stripe offset is 0.

Default: 0

`-split_wire_spacing` *`floating_point_value`*

Specifies the spacing between each split wire.

Default: If you omit this parameter, or set its value to `0`, the software uses the spacing rule in the LEF file.

`-split_wire_weight` *`integer_0_to_10`*

When split wires from a stripe on one layer attempt to make a target connection on another layer but are obstructed by I/O wires, some I/O wires might be deleted so that approximately *n* of 10 split stripe wires complete the target connection.

Note: When you specify this option, the software deletes the fewest I/O wires to accommodate approximately *n* of 10 split stripe wires.

Default: If you omit this parameter, or set its value to `0`, no I/O wires are deleted when the above connections are made.

`-split_wire_width` *`floating_point_value`*

Specifies the width of each split wire.

Default: If you omit this parameter, or set its value to `0`, the software uses the maximum width value in the LEF file.

`-stop_at_last_wire_for_area {0 | 1}`

If set to 1, the stripe antennas are trimmed back to the closest wire or pin of the same net. If set to 0, stripe antennas are not trimmed.

Note: This option is only available if you specify the `-area` parameter of the addStripe command.

Default: `0`

`-stripe_min_length` *`float`*

Specifies the minimum stripe length.

Default: 0.0

`-switch_cellname {`*`cellA cellB ...`*`}`

Treats the specified switch cell as a ring. Pins in the switch cell can then be used as targets for stripes. If the switch cell is in the core area, the software treats it as a block ring. If the switch cell is outside of the floorplan row, the cell is treated as a core ring, and used as the stripe boundary.

You can specify one or more switch cell names.

To turn off this functionality, specify one of the following:

- `-switch_cellname “”`
- `-switch_cellname none`

Default: `none`

`-switch_layer_overlap_length` *`floating_point_value`*

Controls the overlap distance used for the via between a stripe layer and the layer being switched to.

Note: The overlap applies *only* to vias created for straight (non-jogging) connections.

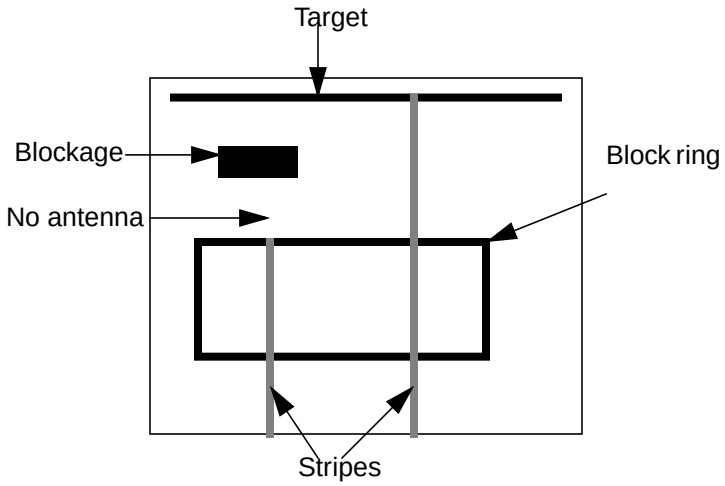
Default: If you omit this parameter, or set its value to `0`, the power planning software creates a square via.

`-trim_antenna_back_to_shape {block_ring | core_ring | pad_ring | stripe | standard_cell_pin}`

Prevents stripes from creating antennas when a wire of the specified shape is the last target before a blockage that prevents connection to another target. Specify one of the following shapes:

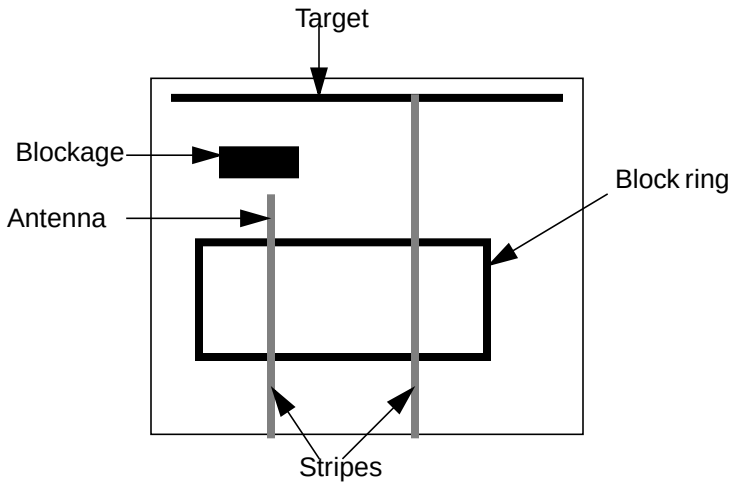
- `block_ring`
- `core_ring`
- `pad_ring`
- `stripe`
- `standard_cell_pin`

The following figure illustrates the software behavior if you specify this parameter with a value of `block_ring`.



Default: `block_ring core_ring`

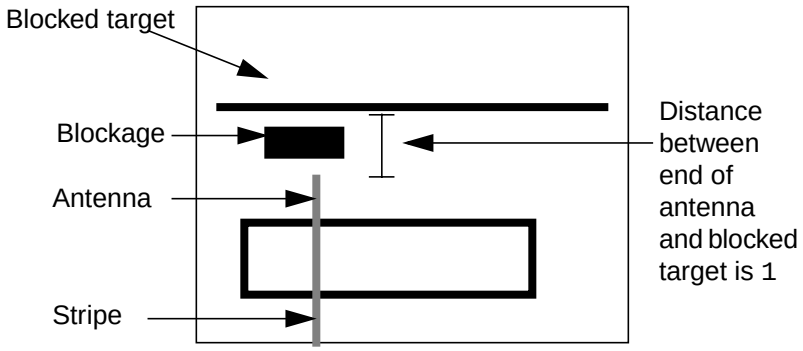
Default: Stripes extend to the blockage, leaving an antenna between the last connection and the blockage. The following figure illustrates the software behavior if you do not specify this parameter.



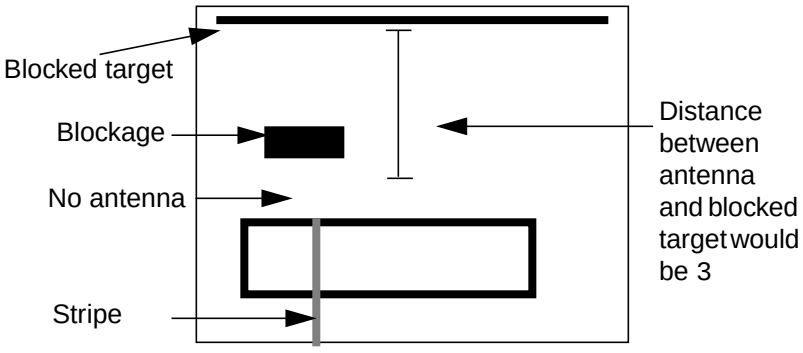
`-trim_antenna_max_distance` *`distance`*

Use this option to specify a distance, in micrometers, between the end of the antenna and a blocked target. This value controls whether an antenna is kept or trimmed for the selected shape. Cadence recommends that you specify a value equivalent to the distance between the core area and the pad ring or core ring.

- If the blocked target is within the specified distance, the antenna is kept, and you can use the SRoute software to complete the connection. For example, if you specify a distance of 2 micrometers and the distance between the end of the antenna and the blocked target is 1 micrometer, the antenna is not trimmed.



- If the blocked target is beyond the specified distance, the antenna is trimmed back to the last connection. For example, if you specify a distance of 2 micrometers and the distance between the antenna and the blocked target would be 3 micrometers, the antenna is trimmed back.



Default: `0.0`

`-use_exact_spacing {0 | 1}`

If set to `1`, the power stripe breaks using the exact spacing rule from the obstructing blockage, pin, or wire to avoid a design rule violation. If set to `0`, the power stripe breaks at 1.5 times the maximum range rule spacing of all layers.

Default: 0

`-use_stripe_width {0 | 1}`

If set to `1`, stripes that jog to connect to a ring always use the stripe width for the jog. If set to `0`, the jog is either the width of the stripe or the width of the ring segment, which ever is smaller.

Note: This parameter only works in conjunction with the following parameter of the `addStripe command:`

- `-jog_to_connect 1`

Default: `1`

`-use_point2point_router {0 | 1}`

If set to `0`, the point-to-point router engine is turned off. If set to `1`, the `addStripe` command automatically calls the internal point-to-point router engine to route the difficult power stripe connections (such as connecting power stripe antennas that were not connected by the `addStripe` command).

Default: `1`

Example

The following command specifies that stripes that jog to connect to a target always use the stripe width.

```
setAddStripeOption -use_stripe_width 1
```

setViaGenOption

```
setViaGenOption
     [-reset_default]
     [-add_pin_to_pin_via {1 | 0}]
     [-create_double_row_cut_via {1 | 0}]
     [-create_via_on_iopin {1 | 0}]
     [-genvia_naming_prefix string]
     [-invoke_verifyGeometry {0 | 1}]
     [-make_via_to {blockWire coreWire followPin IOWire fillWire standardcellPin
     default}]
     [-optimize_cross_via {1 | 0}]
     [-respect_signal_routes {1 | 0}]
     [-skip_via_to {blockWire coreWire followPin IOWire fillWire standardcellPin
     default}]
     [-split_long_via_global_grid {x_offset x_pitch x_length_threshold
     x_length_multiplier y_offset y_pitch y_length_threshold
     y_length_multiplier}]
     [-switch_viarule_direction {1 | 0}]
     [-viarule_preference {predefined | generated | lef_order}]
```

Sets global variables for vias that connect rings and stripes when you use the addRing, addStripe, connectRingPin, `editAddRoute`, `repairPowerWire`, or editPowerVia commands.

Parameters

`-add_pin_to_pin_via {1 | 0}`

Inserts vias to connect pins between different cover macros. To enable this functionality, set this variable to `1`.

Default: `0`

`-create_double_row_cut_via {1 | 0}`

Expands vias from one row cut to two row cuts, provided the following is true:

- Cuts are created within the original intersection area (to connect the two layers)
- Via metal enclosures may overlap outside the intersection area up to where the wires exist (to provide enough of the enclosure over the cuts)

Default: `0`

`-create_via_on_iopin {1 | 0}`

Enables or disables the generation of vias on I/O pins only. To disable via generation for I/O pins, set this variable to `0`.

Default: `1`

`-genvia_naming_prefix` *`viaNamePrefix`*

Sets the specified prefix name as the prefix for all generated vias created by the Encounter software.

Default: none

`-invoke_verifyGeometry {0 | 1}`

Enables or disables the `verifyGeometry` command called by the via generation software.

When set to `1`, the via generation software calls the `verifyGeometry` command to detect any `minStep` violations, then calls the `fixMinStepVia` command to fix them.

Note: This may negatively impact run-time performance.

When set to `0`, the via generation software will not call the `verifyGeometry` command.

Note: This may leave `minStep` violations in the design.

Default: `0`

`-make_via_to {blockWire coreWire followPin I/OWire fillWire standardcellPin default}`

Allows vias to be generated for the specified types of pins and wires.

Default: `default`

`-optimize_cross_via {1 | 0}`

Controls the creation of additional via cuts.

The via generation software uses two methods to create additional via cuts:

- Allows via-metal enclosures to overlap existing metal wires outside the intersection area for small vias (no rows of cuts, or one row of cuts).
- Shrinks large via metals (two or more rows of cuts) to create minimum enclosure.

On the middle layer of stacked vias, the via metal can be expanded up to the via rule enclosure value (along the preferred direction of the layer).

At a wire end, the via metal can be expanded up to the via rule enclosure value (along the wire direction). This allows more cuts to be fit into T-junctions, in the same manner as at crossovers.

Default: `0`

`-reset_default`

Resets all parameters to their default values.

`-respect_signal_routes {1 | 0}`

Controls whether the via generation software honors pre-existing signal routes.

If set to `1`, instructs the via generation software to read pre-existing signal routes and to avoid creating DRC violations for the routes.

Note: Using this setting can affect software performance significantly.

If set to `0`, instructs the via generation software to ignore any pre-existing signal routes.

Note: DRC violations can occur if you use this setting.

Default: `0`

`-skip_via_to {blockWire coreWire followPin I/OWire fillWire standardcellPin default}`

Prevents vias from being generated on the specified types of pins and wires.

Default: `default`

`-split_long_via_global_grid {x_offset x_pitch x_length_threshold x_length_multiplier y_offset y_pitch y_length_threshold y_length_multiplier}`

Provides options to control global grids so that vias on the grids are in line on x and/or y directions.

- `x_offset, y_offset`: specifies in microns the offset from design boundry
- `x_pitch, y_pitch`: specifies in microns the centre to centre distance in split vias
- `x_length_threshold, y_length_threshold`: specifies vias longer or equal to threshold are being split
- `x_length_multiplier, y_length_multiplier`: specifies unit via length multiplier

Default: default

`-switch_viarule_direction {1 | 0}`

Switches the direction of the `HORIZONTAL OVERHANG` and `VERTICAL OVERHANG` of the `LEF VIARULE DIRECTION` statement.

Note: If you issue the `setViaGenOption` command with this parameter, you must do so before you issue the `loadLefFile` command (or any other command that uses the `loadLefFile` command, such as `restoreDesign` or `loadConf`).

Default: `0`

`-viarule_preference {predefined | generated | lef_order}`

Sets the preferred via-rule usage for multiple layer via-rule pairs defined in the LEF technology file. Use this parameter only once before running the `fcroute`, `sroute`, or power planning commands.

- `predefined`
  Uses the predefined via rule defined in the LEF technology file as the preferred usage.
- `generated`
  Uses the generated via rule defined in the LEF technology file as the preferred usage.
- `lef_order`
  Uses the first via rule defined in the LEF technology file as the preferred usage.

Default: `default`

snapRoute

`snapRoute`

Snaps off-grid routes to the manufacturing grid. Off-grid routes include both Manhattan or 45-degree routes and can be generated externally by the Spectra router. Use this command after you perform power routing on the design. Use the `snapRoute` command before running `splitRoute` to snap routes to the manufacturing grid.

Parameters

None

splitRoute

```
splitRoute
     [-absWidth value1]
     [-maxWidth value2]
     [-minSpacing value3]
     [-net netNames]
     [-selected {0 | 1}]
```

Splits wires that exceed the maximum width defined in the LEF file. Divides wide routes into multiple segments using the options for maximum width and spacing. Use this command after you perform power routing on the design. Use the `snapRoute` command before running `splitRoute` to snap routes to the manufacturing grid.

Parameters

`-absWidth` *`value1`*	Specifies the exact width of the split wire, in micrometers.
`-maxWidth` *`value2`*	Specifies the maximum width of the split wire, in micrometers.
`-minSpacing` *`value3`*	Specifies the minimum spacing, in micrometers, between split wire segments.
`-net` *`netNames`*	Specifies the *`netNames`* of all nets you would like to split.
`-selected 0 | 1`	Selects the wires and nets to split using Encounter. Using this option requires clicking on the wires and nets in your design, then issuing this parameter. *Default*: 0

synthesizePowerPlan

```
synthesizePowerPlan
     -totalPower milliwatts
     [-instancePower fileName]
     [-irDropThreshold volts]
     [-templateParameterFile fileName
     [-template templateName [-templateType {DESIGN | IP}]]
```

Important

The `synthesizePowerPlan` command is obsolete. Use the `addStripe` and `addRing` commands to create power structures.

Creates power structures based on the total power value or instance power file, the contents of the power plan template, and the template parameter file for use in prototyping power plans. Use this command after creating an initial floorplan.

A template contains information on

- Core ring layers
- Stripe layers
- Pad rings (optional)
- Option set name

  Create the option set using the Edit Power Planning Option form. For details see Edit Power Planning Option in the “Power Menu” chapter of the *Encounter Menu Reference*.

Parameters

`-instancePower` *`fileName`*

Specifies the instance power file instead of a single total power value (`-totalPower`) to improve the accuracy of IR-drop analysis.

`-irDropThreshold` *`volts`*

Specifies the IR drop threshold for the design.

`-template` *`templateName`*

The name of a power plan template that contains information for creating a power plan topology structure.

Important

Cadence recommends that you use the Edit Power Plan Template to create the power plan template. *templateName* is the name you specify when you save the power plan topology in the Edit Power Plan Template form.

`-templateParameterFile` *`fileName`*

The name of a file that contains parameter information for the design template.

Template parameters include the following data for rings and stripes:

- Width
- Spacing
- Offset
- Pitch

Important

Cadence recommends that you save the template parameters through the Specify Template Parameter form.

`-templateType {DESIGN | IP}`

Identifies the power plan template *templateName* as either a design template or as an IP library template.

Default: If you do not specify this parameter, the power planning software assumes that *templateName* is a design template.

`-totalPower` *`milliwatts`*

Specifies the total average power value for the design.

Example 1

The following command analyzes floorplan and library data to define a template, and estimates the template parameters based on a total power value of 100 mW. After the command estimates the template parameters, it creates the power plan with the Encounter power planning commands:

```
synthesizePowerPlan -totalPower 100
```

Example 2

The following command performs the same steps as in Example 1, but in addition, analyzes power and adjusts the template parameters to create a power plan that meets the given IR drop threshold target:

```
synthesizePowerPlan -totalPower 100 -irDropThreshold 0.05
```

Example 3

The following command reads the custom template `myTemplate` (provided by the user), and defines templates for any blocks that do not have templates. The `synthesizePowerPlan` command then estimates the parameters based on the total power and IR drop threshold values. the command creates a power plan using the template and estimated parameters and analyzes the power. Based on the feedback from power analysis, the command adjusts the parameters and recreates the power plan to meet the IR drop threshold target.

```
synthesizePowerPlan -totalPower 100 -template myTemplate -irDropThreshold 0.05
```

Example 4

In this example, the `synthesizePowerPlan` command creates the power plan that is described in `myTemplate` using the template parameters described in `myTemplateParams.tParam`.

```
synthesizePowerPlan -template myTemplate -templateParameterFile
myTemplateParams.tParam
```

toggleCutRowSelection

`toggleCutRowSelection`

Specifies which type of rows you can select in the design display area. Use this command after you run the cutCoreRow command. The software generates the following two types of rows:

- Rows that flood the core
- Rows generated by inputting the DEF file or by issuing the cutCoreRow command

When used, `toggleCutRowSelection` issues one of the following messages:

- `Cut core rows selection is now enabled`

  This enables you to select rows generated by inputting the DEF file or by issuing the `cutCoreRow` command.

- `Cut core rows selection is now disabled`

  This enables you to select rows that flood the core.

Parameters

None

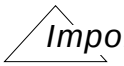
26

Power Route Commands

extractPadRingNetFromNet

`extractPadRingNetFromNet` *`netname`*

Finds the pins for the specified signal net that is located on pad cells. You can then use the `sroute` command to create a pad ring that connects those pins. Use this command after importing the design.



Command `extractPadRingNetFromNet` is obsolete and has not been replaced.

Parameters

`netname`	Specifies the signal net name for which the software creates pad rings. **Note:** If the netname contains a non-alphanumeric character, you must enclose it in curly braces.

fcroute

```
fcroute
     -type {power | signal}
     [-designStyle {aio | pio}]
     [-nets {list_of_net_names}]
     [-routeStyle {manhattan | 45DegreeRoute}]
     [-doubleBendRoute]
     [-shieldBump]
     [-layerChangeBotLayer layerNum]
     [-layerChangeTopLayer layerNum]
     [-jogControl {preferWithChanges | preferSameLayer | preferDifferentLayer}]
     [-straightConnections [straightWithDrcClean] [straightWithChanges]]
     [-routeWidth real]
     [-preventViaUnderBump]
     [-area {x1 y1 x2 y2}]
     [-connectInsideArea]
     [-deleteExistingRoutes]
     [-verbose]
     [-msgRate int]
     [-extraConfig fileName]
     [-differentialRoute]
     [-differentialPairRoute]
     [-differentialRouteTolerance value]
     [-balancePairThreshold percentage]
     [-shieldLayers {a | b | c}]
     [-shieldNet {tieHigh | tieLow}]
     [-shieldWidth userunit]
     [-widthLimit userunit]
     [-splitGap userunit]
     [-interleaveStyle]
     [-constraintFile filename]
     [-connectPowerCellToBump]
     [-minEscapeDistance unit]
     [-globalOnly]
     [-overflowMap]
     [-optWidth]
     [-multiBumpsToPad]
     [-multiPadsToBump]
```

Specifies that power routing and signal routing recognize the bumps specified in a flip chip design.

The command supports all the LEF 5.7 syntax to determine which power/ground pin shape on the I/O driver cell must be connected to a bump. For more information, see the “Performing Area I/O Placement” in the Data Preparation chapter of the *Encounter User Guide*.

Use this command after importing the design.

You can perform power routing or signal routing (`-type`) using the area I/O or peripheral I/O routing styles. Each routing style supports a specific set of `fcroute` command parameters. The following table lists the routing style support for each parameter.

fcroute Parameters	**-designStyle**	
	aio	**pio**
-area {*x1 y1 x2 y2*}	**X**	
-balancePairThreshold *percentage*	**X**	
-connectInsideArea	**X**	
-connectPowerCellToBump	**X**	**X**
-constraintFile filename	**X**	**X**
-deleteExistingRoutes	**X**	**X**[1]
-differentialPairRoute	**X**	
-differentialRoute	**X**	
-differentialRouteTolerance *value*	**X**	
-doubleBendRoute	**X**	**X**
-extraConfig *fileName*	**X**	**X**
-globalOnly		**X**
-interleaveStyle	**X**	**X**
-jogControl {preferWithChanges | preferSameLayer | preferDifferentLayer}	**X**	
-layerChangeBotLayer *layerNum*	**X**	**X**
-layerChangeTopLayer *layerNum*	**X**	**X**
-minEscapeDistance *unit*	**X**	**X**
-msgRate *int*	**X**	**X**
-multiBumpsToPad		**X**
-multiPadsToBump		**X**
-nets {*list_of_net_names*}	**X**	**X**[2]
-optWidth		
-overflowMap		**X**

fcroute Parameters	-designStyle aio	-designStyle pio
-preventViaUnderBump	X	X
-routeStyle {manhattan | 45DegreeRoute}	X	X
-routeWidth *real*	X	X
-shieldBump	X	X
-shieldLayers {a | b | c}	X	
-shieldNet {tieHigh | tieLow}	X	
-shieldWidth *userunit*	X	
-splitGap *userunit*	X	X
-straightConnections [straightWithDrcClean] [straightWithChanges]	X	X
-type {power | signal}	NA	NA
-verbose	X	X
-widthLimit *userunit*	X	X

1. When using the -deleteExistingRoutes parameter in conjunction with the -nets *names* parameter, the -deleteExistingRoutes parameter works only in area I/O mode.
2. When using the -nets *list_of_net_names* parameter in conjunction with the -deleteExistingRoutes parameter, the -nets *list_of_net_names* parameter works only in area I/O mode.

Parameters

-area {*x1 y1 x2 y2*}

Specifies the coordinates for area I/O routing. You must enclose the x and y coordinates in curly braces and separate them with a space.

Note: Encounter issues a warning message if you specify area constraints in the flip chip route PIO mode. (fcroute -designStyle pio)

`-balancePairThreshold` *`percentage`*

The maximum percentage over a direct route above which balanced routing will not occur. For example, if the length is 10 and you do not want to increase it if the new length would be 11 or greater, specify a value of `10`.

Default: `20`

`-connectInsideArea`

Specifies that connections from all sources within the specified area can connect only to targets that are also inside that area.

Note: This parameter is only available if you also specify the `-area` parameter.

Default: If you do not specify this option the software makes connections from all sources within the specified area to targets both inside and outside the specified area.

`-connectPowerCellToBump`

Connects all power bumps to the I/O cell pin. It does not connect to a power or ground stripe or ring. When used with the `-type signal` parameter, you can connect power bumps to I/O power pads. To make this type of connection, you need the following information:

- The Verilog netlist must contain the I/O power pad, for example:

  ```
  VDDCELL VDD_INST (.PAD());
  ```

- The LEF macro pin must contain the `USE POWER` statement, for example:

  ```
  MACRO VDDCELL
  PIN PAD
  USE POWER;
  ```

To improve the quality of routing in situations where there may be many power bumps in the design, you can pair power bumps to power cells by using this parameter in conjunction with the `-constraintFile` parameter. For example:

```
fcroute -connectPowerCellToBump -constraintFile
pair.const
```

where `pair.const` specifies the following information:

```
### Syntax: net pad [pad2] bump [bump2] ###
PAIR
    VDD IOPADS_INST/esd Bump_90_9_8
END PAIR
```

`-constraintFile` *`filename`*

Specifies the name of the file that contains the names of the nets that are to have differential routing. If you do not specify this parameter and if no differential pairs are defined in the PROPERTY statement of the DEF file, differential routing is not done. For more information, see Creating Differential Routing to Signal Bumps in the *Encounter User Guide.*

`-deleteExistingRoutes`

Specifies that the software remove existing connections when you use the `fcroute` command multiple times.
Default: The software maintains existing connections each time you issue the `fcroute` command.

Note: If you specify the `-area` parameter, existing routes are always preserved, even if you also specify the `-deleteExistingRoutes` parameter.

`-designStyle {aio | pio}`

Selects between area I/O and peripheral I/O routing styles.

`-differentialPairRoute`

Retains similar timing delays between two nets that are specified with the `-nets` parameter.

Note: If you want to specify multiple pairs of nets, do so in the routing constraints file.

`-differentialRoute` Specifies that differential routing is done for nets that are not specified in the constraint file or in the PROPERTY statement of the DEF file. If you do not include this parameter, differential routing is done only for nets specified in the constraint file or in the PROPERTY statement of the DEF file.

`-differentialRouteTolerance` *`value`*

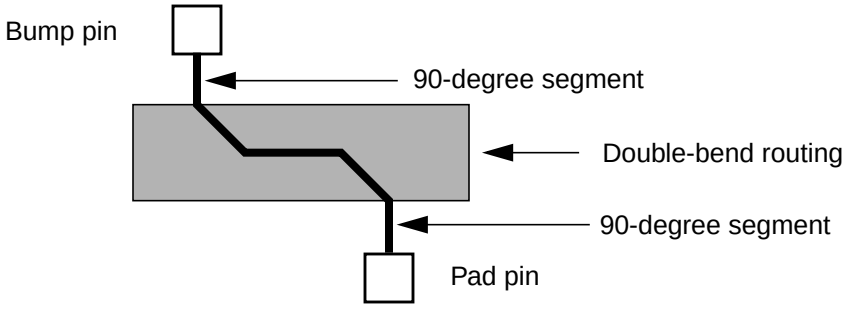
The software measures the length/width ratio of the wires and only performs differential routing if the difference between these length/width ratios is smaller than this value. You can specify a value in the range of `0` to `100`.

For example, if wire A is 10x2, its length/width ratio is 5. If wire B is 8x1, its length/width ratio is 8. The difference between the length/width ratios of these two wires is 3. In order for the software to perform differential routing for these two wires, you would have to specify a value of less than 3.

Note: If the actual difference between the length/width ratios is less than the specified value, the software omits differential routing. If the actual difference is greater than the specified value, the software attempts differential routing, but if the software cannot meet the specified value, the software displays a warning message.

Default: 2

`-doubleBendRoute` Allows double-bend routing on segments between a bump pin and a pad pin.



Note: Segments that connect directly to a bump pin or pad pin are always routed at 90 degrees.

Note: If you specify both `-routeStyle 45DegreeRoute` and `-doubleBendRoute`, the software ignores the `-doubleBendRoute` parameter.

Default: Off

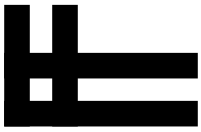
`-extraConfig` *`filename`*

Specifies the name of an extra configuration file. This file contains additional fcroute commands that provide more control over the routes to bumps.

For more information, see Extra Configuration File Options for fcroute on page 1094

`-globalOnly` Executes the global router to provide you with the routability statistics. This is helpful for optimizing mode.

`-interleaveStyle` If you specify this parameter, the split wires interleave with one another, as shown in the following illustration:



If you do not specify this parameter, the split wires do not have an interleaving pattern. The following illustration shows the pattern for the split wires.

`-jogControl {preferWithChanges | preferSameLayer | preferDifferentLayer}`

Specifies that jogs are allowed during routing to avoid DRC violations.

You must specify either the `-jogControl` parameter or the `-straightConnections` parameter.

If you specify the `-jogControl` parameter, you can also specify one of the following options:

`preferWithChanges`

Specify this option if you prefer the software to make straight connections between targets, and to change layers instead of jog to avoid DRC violations. If you do not specify this option, the route uses both layer changes and jogging to avoid DRC violations.

`preferSameLayer`

If the route must jog to avoid a DRC violation, the jog occurs on the same layer whenever possible. This can result in routing in the non-preferred direction.

`preferDifferentLayer`

If the route must jog to avoid a DRC violation, the jog occurs on the layer that is in the preferred routing direction whenever possible.

`-layerChangeBotLayer` *`layerNum`*

Specifies the bottom-most metal layer that the software can use when routing bumps.
Default: If you do not specify this parameter, all layers are used.

`-layerChangeSecondaryBotLayer`

Extends the limit for the bottom-most metal layer that the software can use for routing bumps. This extended limit is used when routing is not possible using only the layers specified with the `-layerChangeBotLayer` parameters.

This option is available only with `fcroute -signal`.

`-layerChangeSecondaryTopLayer`

Extends the limit for the top-most metal layer that the software can use for routing bumps. This extended limit is used when routing is not possible using only the layers specified with the `-layerChangeTopLayer` parameters.

This option is available only with `fcroute -signal`.

`-layerChangeTopLayer` *`layerNum`*

Specifies the top-most metal layer that the software can use when routing bumps.
Default: If you do not specify this parameter, all layers are used.

`-msgRate` *`int`*

Specifies the interval at which progress messages are generated. If you specify `1`, the software generates every progress message; if you specify `2`, the software generates every other progress message, and so on.
Default: `0`

`-minEscapeDistance` *`value`*

Specifies the minimum distance from the edge of the pin to the edge of the wire segment.

`-multiBumpsToPad`

Enables routing from multiple bumps to one pad in parallel.

Note: This parameter is supported by both, `fcroute` AIO and PIO routing styles.

`-multiPadsToBump`

Enables routing from multiple pads to one bump in parallel.

Note: This parameter is supported by both, `fcroute` AIO and PIO routing styles.

`-nets {`*`list_of_net_names`*`}`

Specifies the nets to connect. If you connect to more than one net, you must enclose the net names in curly braces and separate the names with a space.
Default: If you do not specify this parameter, the software connects all power and ground nets in the design.

`-overflowMap`

Displays the thermal map using the width and spacing parameters specified in the extra configuration file.

To display the thermal map in the floorplan view in the GUI, select *All Colors* which opens the Color Preferences form, then select *GCell Overflow*.

Note: The routing congestion can also be displayed using `setLayerPreference` command. (`setLayerPreference gcellOvflow -isVisible 1`)

`-preventViaUnderBump`

Specifies that vias are generated next to the bump instead of directly on the bump.

This option is available only with `fcroute -signal`.

Default: If you do not specify this parameter, vias can be generated that connect directly to the bump.

`-routeStyle {manhattan | 45DegreeRoute}`

Determines the type of routing, either `manahattan` or `45DegreeRoute`.

Default: `45DegreeRoute`

Note: The `-routeStyle 45DegreeRoute` parameter does not work in conjunction with the following parameters:

- `-doubleBendRoute`
- `-shieldNet`

`-routeWidth` *`real`*

Connects wires to bumps using the specified width only, regardless of the size of the bump or the source to which the bump connects.

Default: When using the peripheral I/O routing style, FCroute determines the width of the route as the bump width or I/O cell pin width that is connected to the bump, whichever is smaller.

When using the area I/O routing style, FCroute does not restrict the size of the RDL route to the width of the I/O cell pin and can route a wire that is larger in width than the specified I/O cell pin width.

`-shieldBump`

Specifies the layer where shielding is created in relation to bumps as specified by the `-shieldLayers` option. The default is on the common layer containing the bumps.

`-shieldLayers {a | b | c}`

Specifies where shield layers are created:

- a = *above* the layer containing bumps
- b = *below* the layer containing bumps
- c= on the layer containing bumps ("*common* layer")

Default: c

`-shieldNet {tieHigh | tieLow}`

Specifies that shield wires are created for the nets specified in the constraint file. You must specify either `tieHigh` to indicate that the shield wires are part of the tie high net or `tieLow` to indicate that the shield wires are part of the tie low net. If there is more than one power or ground net in the design, the software chooses one in the SPECIALNETS section to use.

Default: `tieHigh`

If you do not specify this parameter, shield wires are not created for nets, even if they are specified in the constraint file or in the PROPERTY statement of the DEF file.

Note: Shielding is not supported for 45-degree routes using the `-routeStyle 45DegreeRoute` parameter. Shielding only supports manhattan routes using the `-routeStyle manhattan` parameter.

`-shieldWidth` *`userunit`*

Shield wires are created with the specified width. If you specify the `-shieldNet` parameter and do not specify this parameter, shield wires are created with the default minimum wire width.

`-splitGap` *`userunit`* Specifies the minimum distance between split wire segments. If you do not specify this parameter, the distance between split wire segments is the default minimum spacing value that does not cause DRC violations.

`-straightConnections [straightWithDRCClean] [straightWithChanges]`

Specifies that only straight connections are made between targets and that jogs are not allowed during routing.

This option is available only with `fcroute -power`.

You must specify either the `-straightConnections` parameter or the `-jogControl` parameter.

If you specify the `-straightConnections` parameter, you can also specify one or both of the following options:

`straightWithDRCClean`

Leaves a route open if a straight connection cannot reach a target without causing a DRC violation. If you do not specify this option, the software makes a straight connection to a target, even if the route creates a DRC violations.

`straightWithChanges`

Permits the route to change to another layer to avoid DRC violations. If you do not specify this option, the software makes only straight connections on the same layer.

`-type {power | signal}`

Specifies whether power or signal bumps are to be routed.

`-verbose` Specifies that each progress message appears in the log file and the Encounter console.

`-widthLimit` *`userunit`*

Specifies that wires are split and that the width of each wire after the split is less than the specified limit. If you do not specify this parameter and wires must be split, the width of each split wire is the value in the LAYER statement of the LEF file. If LAYER rules are not specified, no splitting occurs.

Extra Configuration File Options for fcroute

You should only use the extra configuration file if you are familiar with its use.

The following variables can be used in the extra configuration file:

- `srouteConnectToAnyOfBump [TRUE | FALSE]`

  The default is `false`.

  Connects a wire to any of the bumps.

- `srouteConnectToCenterOfPin [TRUE | FALSE]`

  The default is `false`.

  Connects a wire to the center of the pad pin.

- `srouteConnectToEdgeOfBump [TRUE | FALSE]`

  The default is `false`.

  Connects a wire to the edge of the bump.

- `srouteFcReportRes` *`fileName`*

  Wrties the resistance report into the specified file using the resistance constraint MAXRES and the inputs.

- `srouteGrouteClusterRegions [Boundary boxes for each cluster region]`

  Allows you to specify bounding box for each pad cluster.

  Default: " "

- `srouteGrouteLengthDriven [TRUE | FALSE]`

  The default is `false`.

  When optimizing placement and bump assignment, bias to vertical / horizontal connection instead of 45-degree connection.

- `srouteGrouteOptimizeWidth [TRUE | FALSE]`

  The default is `false`.

  Note: You must use this option in conjunction with the

`srouteGrouteOptimizeSpacing` option and they cannot be set to `true` at the same time.

Automatically adjusts the width within the max and min constraint. When set to `true`, this configuration file option calls the global router to get the optimized width for each net and writes the results to a constraint file called `width.cons`.

You can create your own constraint file to specify how the width is calculated. For example, if you have 10 nets in the design (`n1` through `n10`), you can create the following constraint file:

```
NETS
   MINWIDTH 2.0
   MAXWIDTH 5.0
   WIDTHSTEP 0.1
   n1 n2 n3
END NETS
NETS
   MINWIDTH 5.0
   MAXWIDTH 10.0
   WIDTHSTEP 0.5
   n4 n5 n6
END NETS
NETS
   WIDTH 3.0
   n7 n8
END NETS
WIDTH 12
```

where:

`n1, n2, and n3`	Have a width between 2 and 5, and an incremental size of 0.1.
`n4, n5, and n6`	Have a width between 5 and 10, and an incremental size of 0.5.
`n7 and n8`	Have a fixed width of 3.0.
`n9 and n10`	Have a fixed width of 12.0.

Note: All the nets within the same NET group will use the same final width. If you allow the nets to have different widths, while satisfying the min and max value, you can use the following configuration file option:

```
srouteGrouteUniformWidth false
```

- `srouteGrouteOptimizeSpacing [TRUE | FALSE]`

  The default is `false`.

  Note: You must use this option in conjunction with the `srouteGrouteOptimizeWidth` option and they cannot be set to `true` at the same time.

  Automatically adjusts the spacing within the max and min constraint. When set to `true`, FCroute calculates the maximum allowable spacing (for the given net width) and writes it to the log file. All spacing values are the same.

- `srouteGrouteSerialBumpRouting [TRUE | FALSE]`

  Allows bumps of the same net to be connected together.

  The default is `false`.

- `srouteGrouteUniformWidth [TRUE | FALSE]`

  All nets in the same width group have uniform width.

  The default is `true`.

- `srouteJogControl [TRUE | FALSE]`

  Allows jogs during routing to avoid DRC violations.

  The default is `false`.

- `srouteLengthLimit [integer value]`

  Specifies the maximum routing wire length for each flip chip net.

  The default is `0`.

- `srouteMinLength [integer value]`

  Specifies the minimum segment length.

  The default is `0`.

- `srouteOutputFailedResistanceGroute [TRUE | FALSE]`

  Displays nets that failed maximum resistance with different colors.

  The default is `false`.

- `sroutePinSpacing` *`dbUnitValue`*

  Defines the I/O pad pin spacing.

- `sroutePowerBumpAllDir [TRUE | FALSE]`

  Connects power bump to power stripes on all directions.

  The default is `false`.

- `sroutePrevent45ForLowerLayer [TRUE | FALSE]`

  Prevents 45-degree routes for lower layer.

  The default is `false`.

- `sroutePreventViaUnderBump [TRUE | FALSE]`

  Prevents via under a bump.

  The default is `false`.

- `sroutePushAndShove [TRUE | FALSE]`

  If `true`, detail routing gets ripped and rerouted, to route open nets.

  The default is `true`.

- `sroutePushAndShoveVerbose [TRUE | FALSE]`

  If `true`, displays debugging messages during `push_and_shove`.

  The default is `false`.

- `srouteReduceLayerChanges` *`integer`*

  The default is `0`.

- `srouteRouteSpacing [integer value]`

  Specifies the default routing space for each flip chip net.

  The default is 0.

- `srouteRouteWidthForLowerLayer` *`dbUnit`*

  The default is `false`.

- `srouteUseSpecifiedWidthForTopLayer {true | false}`

  The default is `false`.

Related Topics

- *Flip Chip Route – Advanced – Routing Constraints GUI* in "Route Menu" chapter of the *Encounter Menu Reference*.
- Routing and Placement Constraints in "Flip Chip Methodologies" chapter of the *Encounter User Guide*.

fixMinCutVia

`fixMinCutVia`

Replaces power vias flagged by a violation marker due to a violation of the LEF `MINIMUMCUT` rule. The via is replaced with a via that contains the minimum number of cuts based on the LEF rule, and the violation marker is removed. If the via cannot be replaced, the violation marker is not removed. Use this command after running the verifyGeometry command.

Parameters

None

fixMinStepVia

```
fixMinStepVia
```

Replaces power vias flagged by a violation marker due to a violation of the LEF `MINIMUMSTEP` rule. The via is replaced with a via that contains the minimum number of steps based on the LEF rule, and the violation marker is removed. If the via cannot be replaced, the violation marker is not removed. Use this command after running the verifyGeometry command.

Parameters

None

routePointToPoint

```
routePointToPoint
     -help
     -bump {target:pad targetLayers:layerNumber[:layerNumber] jogPitch:value}
     -constraintFile filename
     -net floatNetName
     -routeLayer {bot[:top[:step]] [,bot[:top[:step]]]}
     -routeNets {netNames}
     -routePoints {pinOptions [instaName | "-1"] [pinName | "-1"] [location]
      | netOptions netName [layer | "-1"] location}
     -routeStyle ["manhattan" , "doubleBend" , "diagonal"]
     -spacing spacingValue
     -split {layerBase maxWidth:value [gap:value] [style:RIVER|MASH]}
     -stripe {target:pad targetLayers:layerNumber[:layerNumber] jogPitch:value}
     -width widthValue
```

Specifies routing constraints to perform point-to-point routing between I/O pad pins and bumps, and wires and bumps for SPECIALNETS (only) that are defined in the DEF file.

Parameters

`-bump {target:`*`pad`* `targetLayers:`*`layerNumber`*`[:`*`layerNumber`*`] jogPitch:`*`value`*`}`

Specifies routing between a bump and I/O pad, with tapering pin widths.

For example, `-bump {target:pad targetLayers:M7:M7 jogPitch:160}`

`-constraintFile` *`fileName`*

Specifies the name of the file that contains the nets defined in the DEF file.

`-net` *`floatNetName`*

Specify the wire net name.

`-net netOption`

Specifies the net options such as the net name, net layer, or the location coordinates.

`-pin pinOption`

Specifies the pin options such as the instance name, the pin name or the location coordinates.

`-routeLayer {bot[:top[:step]] [,bot[:top[:step]]]}`

Specifies the route layer.

`-routeNets {`*`netNames`*`}`

Specifies the net names for routing.

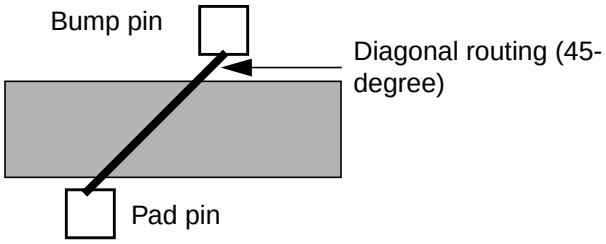
`-routePoints {pinOptions | netOptions}`

Specifies two routing points.

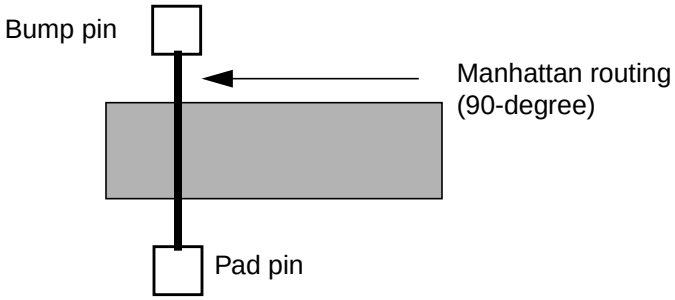
`-routeStyle [“manhattan , “doubleBend” , “diagonal”]`

Specifies the routing style options for performing the point-to-point routing.

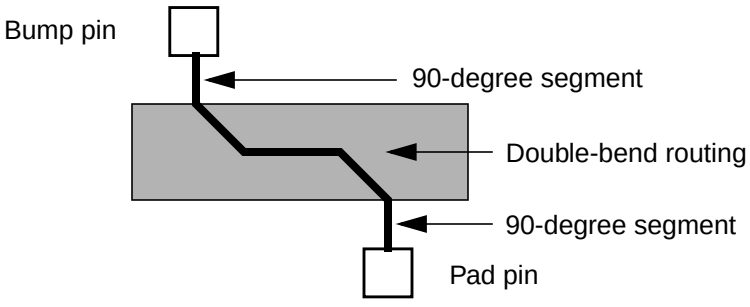
- `diagonal`



- `manhattan`



- `doubleBend`



`-spacing` *`spacingValue`*

Specifies the route spacing, in micrometers.

`-split {layerBase maxWidth:value [gap:value] [style:RIVER|MASH]}`

Splits the wire segment based on the maximum width and splitting style.

The router gets the maximum width value from the layer information; the router does not split the wire segment when the wire width does not exceed the layer's maximum width.

- `maxWidth:value`

  Specifies the maximum width for splitting.

- `gap:value`

  Specifies the splitting gap.

- `style:[RIVER|MESH]`

  Specifies the splitting style:

  `RIVER`: Allows only one crossing of the split wire.

  `MESH`: Allows crossing of all of split wires and also drops all the crossing sections.

`-stripe {target:`*`pad`* `targetLayers:`*`layerNumber`*`[:`*`layerNumber`*`] jogPitch:`*`value`*`}`

Specifies point-to-point routing between the end of stripe and the target I/O pad pin within the specified jog pitch; The target pins can have tapering pin widths.

Example

```
routePointToPoint -routeLayer M7:M7 -width 35 -
spacing 2.0
-routeStyle diagonal
-stripe {target:pad targetLayers:M7:M7
jogPitch:160}
-constraintFile ../p2pConstraint.const -verbose
```

`-width` *`widthValue`*

Specifies the route width, in micrometers.

Examples

The following command performs diagonal style point-to-point routing between the I/O pad pin `newIo_2` and bump `Bump_41_0_4` on route layer `M8`, having a default route width of `0.44` micrometers, and a spacing of `0.46` micrometers:

```
routePointToPoint -routeLayer M8:M8 -width 0.44 -spacing 0.46 -routeStyle diagonal
-pin {newIo_2 -1 (-21.3115 535.932)} -pin {Bump_41_0_4 port_pad_data_in[10] (76.507
482.032)}
```

Related Topics

- Route Menu chapter in the *Encounter Menu Reference*
  - Point-To-Point Route
- Flip Chip Methodologies chapter in the *Encounter User Guide*
  - Point-To-Point Routing
- Flip Chip Methodologies chapter in the *Encounter User Guide*
  - Routing Constraints

sroute

```
sroute
     [-allowJogging {0 | 1}]
     [-allowLayerChange {0 | 1}]
     [-area x1 y1 x2 y2]
     [-block {names_of_blocks}]
     [-blockPin {useLef | all | onBoundary | {leftBoundary | rightBoundary |
        topBoundary | bottomBoundary} | abutPins}]
     [-blockPinLayerRange {minLayerNum maxLayerNum}]
     [-blockPinLayerRange {minWidth maxWidth}]
     [-blockPinJogViaMultiplier viaSizeMultiplier]
     [-blockPinRouteWithPinWidth]
     [-blockPinTarget {nearestRingStripe | nearestRing | nearestTarget |
        boundaryWithPin | padRing | farthestPadRing }]
     [-blockPinToAlignedPadPin]
     [-connect {block | corePin | padPin | padRing | floatingStripe}]
     [-connectAlignedBlockAndPadPins {blockPisAsTarget | padPinAsTarget}]
     [-connectInsideArea]
     [-corePinCheckStdcellGeoms]
     [-corePinLayer layerNumList]
     [-corePinMaxViaHeight viaSizePercent]
     [-corePinMaxViaWidth viaSizePercent]
     [-corePinNoRouteEmptyRows]
     [-corePinJogViaMultiplier viaSizeMultiplier]
     [-corePinWidth real]
     [-crossoverViaBottomLayer layerNumber]
     [-crossoverViaTopLayer layerNumber]
     [-deleteExistingRoutes]
     [-extraConfig filename]
     [-layerChangeRange {bottomLayerNum topLayerNum}]
     [-nets {names}]
     [-noBlockPinOneAmongOverlappedPins]
     [-padCellSkipRoutingOnSide {bottom | top | left | right}]
     [-padPinLayerRange {minLayerNum maxLayerNum}]
     [-padPinMinViaSize viaSizePercent]
     [-padPinToAlignedBlockPin]
     [-padPinWidth real]
     [-padPinPortConnect {onePort | allPort | oneGeom | allGeom | preferLayer}]
     [-padRingNoLefConvention]
     [-padRingWidth real]
     [-padRingLayer {layerNumList)]
     [-powerDomains [powerDomainName1 powerDomainName2 …]]
     [-secondaryPinNet {list_of_nets}]
     [-secondaryPinRailVerticlaStripeGrid {topLayerNum width pitch}]
     [-splitLogVia {length spacing offset}]
     [-stripeSCpinTarget {firstAfterRowEnd | boundaryWithPin | padRing |
     farthestPadRing | none }]
     [-stripeJogViaMultiplier viaSizeMultiplier]
     [-targetObjListFile filename]
     [-targetPenetration wiretype percentage]
```

```
[-targetViaBottomLayer layerNumber]
[-targetViaTopLayer layerNumber]
[-viaConnectToShape [blockring] [ring] [padring] [stripe]
     [blockwire] [corewire] [followpin] [iowire] [noshape]
[-verbose]
[-viaThruToClosestRing]

(obsolete parameters)

[-blockPin {useLefConvention | allPins | boundaryPins}]
[-blockPinBoundaryOnly]
[-blockPinConnectRingPinCorners]
[-blockPinMaxLayer layerNumber]
[-blockPinMinLayer layerNumber]
[-blockPinMaxWidth  real]
[-blockPinMinWidth  real]
[-blockPinTarget {routeToRingorStripe | routeToRingOnly}]
[-blocks {all | named -blockNames {list_of_blocks}}]
[-blockSides [top] [bottom] [left] [right]]
[-jogControl [preferWithChanges] {preferSameLayer | preferDifferentLayer}]
[-layerChangeBotLayer layerNumber]
[-layerChangeTopLayer layerNumber]
[-lsStripeLayer layerNumber
[-lsStripeWidth integer]
[-lsStripePitch integer]
[-msgRate int]
[-noLayerChangeRoute]
[-noPadPins]
[-noPadRings]
[-noBlockPins]
[-noPadRings]
[-noStripes]
[-padPinAllGeomsConnect | -padPinPreferredLayerOnly]
[-padPinMinLayer layerNumber]
[-padPinMaxLayer layerNumber]
[-padPinOnePortOnly]
[-padPinSkipBottom]
[-padPinSkipLeft]
[-padPinSkipRight]
[-padPinSkipTop]
[-secondaryStopSCPin {firstStripe | atRowEnd | atCellEnd | noDRC}]
[-shifterPinConnection list_of_nets]
[-split_long_via {threshold_value step_value offset_value}]
[-stopBlockPin {nearestTarget | boundary | boundaryWithPin | padRing
     | lastPadRing}]
[-stopStripeSCPin {firstAfterRowEnd | boundary | boundaryWithPin | padRing
     | lastPadRing | none}]
[-straightConnections [straightWithDrcClean] [straightWithChanges]]
```

Routes power structures. Use this command after creating power rings and power stripes. Depending on the parameters you use, it can do any or all of the following:

- Create pad rings
- Connect pins on specified nets on the blocks
- Connect pads to nearby rings or stripes
- Extend unconnected stripes to make a connection
- Route standard cell pins

Note: To undo the `sroute` command, use the `defOut` command to save the design database before you issue the `sroute` command. Then you can use the `defIn` command to restore the design to its state before running `sroute`.

The command supports all the `LEF 5.7` syntax to determine which power/ground pin shape on the I/O driver cell must be connected to a bump. For more information, see the "Performing Area I/O Placement" in the Data Preparation chapter of the *Encounter User Guide*.

Parameters

`-allowLayerChange {0 | 1}`

Allows connections to targets on different layers.

Default:

Note: This parameter replaces the `-straightConnections` and `-noLayerChangeRoute` parameters.

`-allowJogging {0 | 1)`

Specifies that jogs are allowed during routing to avoid DRC violations.

Default:

Note: This parameter replaces the `-straightConnections`, and `-jogControl` parameters.

`-area` *`x1 y1 x2 y2`* Specifies the coordinates for the area in which power is to be routed.
Default: If you do not specify this parameter, the entire design is routed.

Note: If you specify this parameter, existing routes are always preserved, even if you also specify the `-deleteExistingRoutes` parameter.

`-block {`*`names_of_blocks`*`}`

Specifies the blocks to connect.

Note: This parameter replaces the `-blocks` parameter.

`-blockPin {useLef | all | onBoundary | topBoundary | bottomBoundary} | abutPins}`

Specifies which block pins to connect.

Note: This parameter replaces the `-blockPin {useLefConvention | allPins | boundaryPins}`, `-blockPinBoundaryOnly`, and `-blockSides` parameters.

Use one of the following options:

`abutPins`	Pins are abutted.
`all`	All pins on all ports are connected, regardless of how pins are specified in the LEF file.
`bottomBoundary`	Only pins on the bottom boundary are connected.
`onBoundary`	Connects *only* pins that abut block boundaries.
`topBoundary`	Only pins on the top boundary are connected.
`useLef`	Pins are connected exactly as specified in the LEF file. For example, if the LEF file has multiple ports with one geometry connecting to each port, then one pin per port is connected.

`-blockPin {useLefConvention | allPins | boundaryPins}`

Specifies which block pins are to be connected. Use one of the following options:

`useLefConvention`

Pins are connected exactly as specified in the LEF file. For example, if the LEF file has multiple ports with one geometry connecting to each port, then one pin per port is connected.

`allPins` All pins on all ports are connected, regardless of how pins are specified in the LEF file.

`boundaryPins` Only boundary pins are connected. You can use the `-blockSides` parameter to limit boundary pin connections to specific sides of the block. If you do not specify the `-blockSides` parameter, pins on all sides of the block are connected.

Note: This parameter is obsolete and has been replaced by `-blockPin {useLef | all | onBoundary | {leftBoundary | rightBoundary | topBoundary | bottomBoundary} | abutPins}`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinBoundaryOnly`

Connects *only* pins that abut block boundaries.

Note: This parameter is obsolete and has been replaced by `-blockPin {useLef | all | onBoundary | {leftBoundary | rightBoundary | topBoundary | bottomBoundary} | abutPins}`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinConnectRingPinCorners`

Specifies that the software connects the rings pins to the nearby targets by extensions at the corners. This creates a total of eight possible connections per net. If you do not specify this parameter, the software does not extend the ring pins at the corners.

Note: This option is useful when internal power connections from the ring pin are modeled as obstructions.

Note: This parameter is obsolete and has not been replaced. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinJogViaMultiplier` *`real`*

Specifies a real number to be used as a multiplier that increases the size of vias when wire switches layers. For example, if you specify a value of `1.5`, the via will be one-and-a-half times the size of a full-size via. This can help reduce resistance.
Default: If you do not specify this parameter, the software uses a value of `1`.

`-blockPinMaxLayer` *`layerNumber`*

Specifies the top-most block pin layer to connect to. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth.
Default: If you do not specify this parameter, block pins connect to the highest metal layer in the design.

Note: This parameter is obsolete and has been replaced by `-blockPinLayerRange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinLayerRange`

Specifies the layers on which the tool can make connections, between the bottom-most and top-most layers. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth.

Note: This command replaces the `-blockPinMaxWidth` and `-blockPinMinWidth` commands.

`maxLayerNum` Specifies the bottom-most block pin layer to connect to.

`minLayerNum` Specifies the bottom-most block pin layer to connect to.

`-blockPinMaxWidth` *`real`*

Specifies the maximum block pin width to connect to.
Default: If you do not specify this parameter, the software uses a value of `100`.

Note: This parameter is obsolete and has been replaced by `-blockPinWidthRange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinMinLayer` *`layerNumber`*

Specifies the bottom-most block pin layer to connect to. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth.
Default: If you do not specify this parameter, the software uses a value of `1`, specifying the lowest metal layer in the design.

Note: This parameter is obsolete and has been replaced by `-blockPinLayerRange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinMinWidth` *`real`*

Specifies the minimum block pin width to connect to. The default value is `0.0`

Note: This parameter is obsolete and has been replaced by `-blockPinWidthRange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinRouteWithPinWidth`

Specifies that the block wire used for a connection is to have the same width as the pin from which it connects.
Default: If you do not specify this option, the block wire width is either the width of the target or the width of the pin, whichever is smaller.

`-blockPinTarget {routeToRingorStripe | routeToRingOnly}`

Specifies whether stripes are considered targets for block pin connections.
Default: If you do not specify this parameter, both rings and stripes are considered targets for block pin connections

Specify one of the following options:

`routeToRingorStripe`

The software considers both rings and stripes as targets for block pin connections.

`routeToRingOnly`

The software only considers rings as targets for block pin connections. Block pins are never connected to stripes.

Note: This parameter is obsolete and has been replaced by `-blockPinTarget {nearestRingStripe | nearestRing | nearestTarget | boundaryWithPin | padRing | farthestPadRing}`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinTarget {nearestRingStripe | nearestRing | nearestTarget | boundaryWithPin | padRing | farthestPadRing}`

Specifies the which target to use for routing block pins.

Note: This parameter replaces `-blockPinTarget {routeToRingorStripe | routeToRingOnly}`, `-stopBlockPin`, and extra config variable `srouteBlockToStripe`.

`-blockPinToAlignedPadPin`

Specifies that a pad pin can be considered a connection target for a block pin if the pad pin is aligned with the block pin.
Default: If you do not specify this parameter, pad pins are never considered a target for block pin connection.

Note: This parameter is obsolete and has been replaced by `-connectAlignedBlockAndPadPins`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockPinWidthRange`

Specifies the allowable pin widths, between the maximum and minimum widths given.

This parameter replaces `-blockPinMaxWidth` and `-blockPinMinWidth`.

`maxWidth` Specifies the maximum block pin width to connect to.
Default: If you do not specify this parameter, the software uses a value of `100`.

`minWidth` Specifies the minimum block pin width to connect to. The default value is `0.0`

`-blocks {all | named}`

Controls which blocks are to be connected.
Default: If you do not specify this parameter, all blocks are connected. Specify one of the following:

`all` Connects the power and ground pins of all blocks. This is the default behavior if you do not specify the `-blocks` parameter.

`named` Connects only to blocks that you specify using the `-blockNames` parameter.

`-blockNames {`*`list_of_blocks`*`}`

Specifies the instance names of the blocks to connect to. This parameter must be used with the `-blocks named` parameter. If you specify more than one block instance name, you must enclose the list within curly braces and separate block instance names with a space. To exclude a block from routing, you can add a ~ character at the beginning of the instance name.

Note: This parameter is obsolete and has been replaced by `-block`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-blockSides {[top] [bottom] [left] [right]}`

Determines the connection of block pins when you specify the `-blockPin boundaryPins` option. Specify one or more of the following options:

`top`	Controls connections to boundary pins at the top of the block. *Default*: On
`bottom`	Controls connections to boundary pins at the bottom of the block. *Default*: On
`left`	Controls connections to boundary pins on the left side of the block. *Default*: On
`right`	Controls connections to boundary pins on the right side of the block. *Default*: On

Note: This parameter is obsolete and has been replaced by `-blockPin {useLef | all | onBoundary | {leftBoundary | rightBoundary | topBoundary | bottomBoundary} | abutPins}`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-connect {blockPin | corePin | padPin | padRing | floatingStripe}`

Connects the specified objects to rings and stripes.

This parameter replaces `-noBlockPins`, `-noPadPins`, `-noCorePins`, `-noPadRings`, and `-noStripes`.

`-connectAlignedBlockAndPadPins {blockPinAsTarget | padPinAsTarget}`

Specifies whether block or pad pins can be connection targets for aligned pad or block pins.

Note: This parameter replaces `-blockPinToAlignedPadPin` and `-padPinToAlignedBlockPin`

`blockPinAsTarget`

Specifies that a block pin can be considered a connection target for a pad pin if the block pin is aligned with the pad pin.

`padPinAsTarget`

Specifies that a pad pin can be considered a connection target for a block pin if the pad pin is aligned with the block pin.

`-connectInsideArea` Specifies that connections from all sources within the specified area can connect only to targets that are also inside that area.

Note: This parameter is only available if you also specify the `-area` parameter.

Default: If you do not specify this option the software makes connections from all sources within the specified area to targets both inside and outside the specified area.

`-corePinCheckStdcellGeoms`

Checks the standard cell geometries in the design for any spacing violations between the power rail to power stripe vias and cell blockages to repair any DRC violations. When detected, Encounter repairs these violations by trimming the via arrays as needed. Use this parameter after routing the power structures.

Default: off

`-corePinJogViaMultiplier` *`viaSizeMultiplier`*

Specifies a real number to be used as a multiplier that increases the size of vias in standard cell rails that need to jog and switch layers to connect to the target. For example, if you specify a value of `1.5`, the via will be 1.5 times the size of a full-size via. This can help reduce resistance.
Default: If you do not specify this parameter, the software uses a value of `1`.

`-corePinLayer {`*`layerNumList`*`}`

Connects power pins only on the specified layers. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth. If you specify more than one layer, you must enclose the list in curly braces.
Default: If you do not specify this parameter, the software automatically determines the layer.

`-corePinMaxViaHeight` *`viaSizePercent`*

Controls the length of vias at the crossover of core pin rails and stripes. Specify this value as percentages of the height of the via at the top of the stack. Smaller vias leave room for more routing tracks on layers between the vias.
Default: If you do not specify this parameter, the software uses a value of `100`, which specifies a full-size via.

`-corePinMaxViaWidth` *`viaSizePercent`*

Controls the width of vias at the crossover of core pin rails and stripes. Specify this value as percentages of the width of the via at the top of the stack. Smaller vias leave room for more routing tracks on layers between the vias.
Default: If you do not specify this parameter, the software uses a value of `100`, which specifies a full-size via.

`-corePinNoRouteEmptyRows`

Specifies not to route standard cell rows within empty rows.

`-corePinWidth` *`real`*

Routes only pins that have the specified width.
Default: If you do not specify this parameter, the software automatically calculates the width of the power pins to use. It is usually not necessary to specify a pin width.

`-crossoverViaBottomLayer` *`layernumber`*

Specifies the lowest layer that can be used for via stacking at the crossover point between power structures.

Note: This parameter does not apply to T-pattern connections.

Default: If you do not specify this parameter, all layers are used.

`-crossoverViaTopLayer` *`layernumber`*

Specifies the highest layer that can be used for via stacking at the crossover point between power structures.
Default: If you do not specify this parameter, all layers are used.

`-deleteExistingRoutes`

Specifies that the software remove existing connections when you use the `sroute` command multiple times.
Default: The software maintains existing connections each time you issue the `sroute` command.

Note: If you specify the `-area` parameter, existing routes are always preserved, even if you also specify the `-deleteExistingRoutes` parameter.

`-extraConfig` *`filename`*

Specifies the name of an extra configuration file. This file contains additional sroute commands that provide more control over the power routes.

For information about the options available, see "Extra Configuration File Options for sroute" on page 1135.

`-jogControl [preferWithChanges]`
`{preferSameLayer | preferDifferentLayer}`

Specifies that jogs are allowed during routing to avoid DRC violations.

Note: You must specify either the `-jogControl` parameter or the `-straightConnections` parameter.

If you specify the `-jogControl` parameter, you can also specify the following options:

`preferWithChanges`

Specify this option if you prefer the software to make straight connections between targets, and to change layers instead of jog to avoid DRC violations. If you do not specify this option, the route uses both layer changes and jogging to avoid DRC violations.

`preferSameLayer`

If the route must jog to avoid a DRC violation, the jog occurs on the same layer whenever possible. This can result in routing in the non-preferred direction.

Note: You must specify either `preferSameLayer` or `preferDifferentLayer` when you use the `-jogControl` parameter.

`preferDifferentLayer`

If the route must jog to avoid a DRC violation, the jog occurs on the layer that is in the preferred routing direction whenever possible.

Note: You must specify either `preferSameLayer` or `preferDifferentLayer` when you use the `-jogControl` parameter.

Note: This parameter is obsolete and has been replaced by `-allowJogging and -allowLayerChange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-layerChangeBotLayer` *`layerNumber`*

Specifies the bottom-most metal layer that the software can use when routing power structures.
Default: If you do not specify this parameter, all layers are used.

Note: This parameter is obsolete and has been replaced by `-layerChangeRange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-layerChangeRange`

Allows routing between the specified bottom-most and top-most layer, inclusive.
Default: If you do not specify this parameter, the tool can use any layer.

Note: This parameter replaces `-layerChangeBotLayer` and `-layerChangeTopLayer`.

`bottomLayerNum`

Specifies the bottom-most metal layer that the software can use when routing power structures.

`totpLayerNum`

Specifies the top-most metal layer that the software can use when routing power structures.

`-layerChangeTopLayer` *`layerNumber`*

Specifies the top-most metal layer that the software can use when routing power structures.
Default: If you do not specify this parameter, all layers are used.

Note: This parameter is obsolete and has been replaced by `-layerChangeRange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-lsLayerWidth` *`integer`*

Specifies the width (in user units) of stripes that connect to level-shifter pins.

Default: If you omit this parameter, the software uses the rail width.

Note: This parameter is obsolete and has been replaced by `-secondaryPinRailVerticalStripeGrid`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-lsStripeLayer` *`layerNumber`*

Defines the stripe layer on which level-shifter pins should be connected.

Default: If you omit this parameter, the software uses the preferred layer.

Note: This parameter is obsolete and has been replaced by `-secondaryPinRailVerticalStripeGrid`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-lsStripePitch` *`integer`*

Specifies the pitch (in user units) of stripes that connect to level-shifter pins.

Default: Twice the level-shifter width plus the width value specified in `-lsStripeWidth`.

Note: This parameter is obsolete and has been replaced by `-secondaryPinRailVerticalStripeGrid`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-msgRate` *`int`*
Specifies the interval at which progress messages are generated. If you specify `1`, the software generates every progress message; if you specify `2`, the software generates every other progress message, and so on.

Default: `0`

Note: This parameter is obsolete and has not been replaced. To ensure compatibility with future releases, remove this parameter from your scripts.

`-nets` **{***`names`***}**
Specifies the names of the nets to connect. If you connect to more than one net, you must enclose the net names in curly braces and separate the names with a space.
Default: If you do not specify this parameter, the software connects all power and ground nets in the design.

`-noBlockPinOneAmongOverlappedPins`

Specifies that all pins (including those that overlap) should be routed.

Note: This parameter is effective *only* if `sroute -blockPin allPins` is specified.

Default: If you omit this parameter, if multiple pins overlap either fully or partially, only one will be routed.

`-noBlockPins`
Specifies that power and ground pins of blocks are not connected to rings and stripes.

Note: This parameter is obsolete and has been replaced by `-connect`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-noCorePins`
Specifies that the power and ground pins of the standard cells in the core rows are not connected.

Note: This parameter is obsolete and has been replaced by `-connect`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-noLayerChangeRoute`

Specifies that only targets on the same layer are to be connected.

Note: This parameter is obsolete and has been replaced by `-allowJogging and -allowLayerChange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-noPadPins` Specifies that the power and ground pins of the power pads are not connected into the core power ring.

Note: This parameter is obsolete and has been replaced by `-connect`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-noPadRings` Specifies that pad rings should not be routed. This option is turned off by default.

Note: This parameter is obsolete and has been replaced by `-connect`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-noStripes` Specifies that unconnected stripes should not be extended to allow a connection.

Note: This parameter is obsolete and has been replaced by `-connect`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padCellSkipRoutingOnSide`

Suppresses connection to pad cells on specified sides.

Note: This parameter replaces `-padPinSkipBottom`, `-padPinSkipLeft`, `-padPinSkipRight`, and `-padPinSkipTop`.

`bottom` Specifies that power pads along the bottom side of the chip should not be connected.

`left` Specifies that power pads along the left side of the chip should not be connected.

`right` Specifies that power pads along the right side of the chip should not be connected.

`top` Specifies that power pads along the top side of the chip should not be connected.

`-padPinAllGeomsConnect`

Routes to all geometries of the same port for each pad pin. This parameter is mutually exclusive with the `-padPinPreferredLayerOnly` parameter.
Default: If you do not specify this parameter or the `-padPinPreferredLayer` parameter, only one geometry per port is connected.

Note: This parameter is obsolete and has been replaced by `-padPinPortConnect`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinLayerRange`

Specifies that routing is allowed between the bottom-most and top-most pad pin layers, inclusive.

This parameter replaces `-padPinMaxLayer` and `-padPinMinLayer`.

`maxLayerNum` Specifies the top-most pad pin layer allowed for connection.

`minLayerNum` Specifies the bottom-most pad pin layer allowed for connection.

`-padPinMaxLayer` *`layerNumber`*

Specifies the top-most pad pin layer to connect to. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth.
Default: If you do not specify this parameter, the software connects pad pins to the highest metal layer in the design.

Note: This parameter is obsolete and has been replaced by `-padPinLayerRange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinMinLayer` *`layerNumber`*

Specifies the bottom-most pad pin layer to connect to. A value of `1` indicates the lowest metal later, `2` indicates the next layer up, and so forth.
Default: If you do not specify this parameter, the software uses a value of `1`, which is the lowest metal layer in the design.

Note: This parameter is obsolete and has been replaced by `-padPinLayerRange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinMinViaSize` *`viaSizePercent`*

Permits the software to reduce the size of vias at pad pins when changing layers, if there is not enough room for a standard size via.
Default: If you do not specify this parameter, the software uses a value of `20`, which permits vias that are 20% of the standard via size.

`-padPinPortConnect`

Specifies connections to pad pin ports.

Note: This parameter replaces `-padPinAllGeomConnect`, `-padPinPreferredLayerOnly`, and `-padPinOnePortOnly`.

`allGeom`	Routes to all geometries of the same port for each pad pin.
`allPort`	Routes to all ports.
`oneGeom`	Routes to one geometry per port only.
`onePort`	Routes to only one port of a pad pin if multiple ports are defined in the LEF file.
`preferLayer`	Routes to all geometries on the same port, but in the preferred routing direction only.

`-padPinOnePortOnly`

Specifies that the router connects only one port of a pad pin if multiple ports are defined in the LEF file.
Default: If you do not specify this parameter, all ports are connected.

Note: This parameter is obsolete and has been replaced by `-padPinPortConnect`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinPreferredLayerOnly`

Routes to all geometries on the same port, but in the preferred routing direction only. This parameter is mutually exclusive with the `-padPinAllGeomsConnect` parameter.
Default: If you do not specify this parameter or the `-padPinAllGeomsConnect` parameter, only one geometry per port is connected.

Note: This parameter is obsolete and has been replaced by `-padPinPortConnect`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinSkipBottom` Specifies that power pads along the bottom of the chip should not be connected.

Note: This parameter is obsolete and has been replaced by `-padcellSkipRoutingOnSide`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinSkipLeft` Specifies that power pads to the left of the chip should not be connected.

Note: This parameter is obsolete and has been replaced by `-padcellSkipRoutingOnSide`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinSkipRight`

Specifies that power pads to the right of the chip should not be connected.

Note: This parameter is obsolete and has been replaced by `-padcellSkipRoutingOnSide`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinSkipTop`

Specifies that power pads along the top of the chip should not be connected.

Note: This parameter is obsolete and has been replaced by `-padcellSkipRoutingOnSide`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinToAlignedBlockPin`

Specifies that a block pin can be considered a connection target for a pad pin if the block pin is aligned with the pad pin.
Default: If you do not specify this parameter, block pins are never considered a target for pad pin connection.

Note: This parameter is obsolete and has been replaced by `-connectAlignedBlockAndPadPins`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-padPinWidth` *`real`*

Routes only pad pins that have the specified width. If no width is specified, the software automatically calculates the width of the pad pins to use. It is usually not necessary to specify a pin width.

`-padRingLayer` *`layerNumber`*

Specifies which routing layer to use. A value of 1 indicates the lowest metal later, 2 indicates the next layer up, and so forth. The default is all layers.

`-padRingNoLefConvention`

Specifies that the software chooses the pad pin connection style (`ABUTMENT` or `FEEDTHRU`) when routing pad rings, disregarding what is specified in the LEF file.
Default: If you do not specify this parameter, pad rings are routed based on what is specified in the LEF file.

`-padRingWidth` *`real`*

Specifies the width of pins to connect with. By default, the software automatically calculates this width.

`-powerDomains [`*`powerDomainName1 powerDomainName2 …`*`]`

Restricts `sroute` activity to the specified power domains.

Default: All power domains

`-secondaryPinNet {`*`list_of_nets`*`}`

Connects voltage level shifter pins on the specified nets to the closest segment of the ring around the power domain. This option is available only if a power domain is defined.

Note: This parameter replaces `-shifterPinConnection`.

`-secondaryPinRailVerticalStripeGrid {topLayerNum width pitch}`

Specifies connections to secondary power pins.

Note: This parameter replaces `-lsStripeLayer`, `-lsStripeWidth`, and `-lsStripePitch`.

`topLayerNum`	Specifies the top-most stripe layer on which the tool can route secondary power pins.
`width`	Specifies the width (in user units) of stripes that connect to secondary power pins.
`pitch`	Specifies the pitch (in user units) of stripes that connect to secondary power pins.

`-secondaryStopSCPin {firstStripe | atRowEnd | atCellEnd | noDRC}`

Specifies the secondary extension target for standard cell pins if the primary extension target specified by the `-stopStripeSCPin` parameter is not available.
Default: If you do not specify this parameter, standard cell pins extend to the end of the core row when the specified primary target is not available.

Specify one of the following:

`firstStripe`	Extends the standard cell pin to the first stripe after the last cell in the core row.
`atRowEnd`	Extends the standard cell pin to the end of the core row.
`atCellEnd`	Extends the standard cell pin to the last cell in the core row.
`noDRC`	Extends the standard cell pin as far as possible without creating a DRC violation.

Note: This parameter is obsolete and has not been replaced. To ensure compatibility with future releases, remove this parameter from your scripts.

`-shifterPinConnection {`*`list_of_nets`*`}`

Connects voltage level shifter pins on the specified nets to the closest segment of the ring around the power domain. This option is available only if a power domain is defined.

Note:

`-split_long_via {`*`threshold_value step_value offset_value`*`}`

Splits vias that are longer than the specified length into smaller vias that have the specified center-to-center step and bottom/left edge offset (all values in micrometers).

Default: If you do not specify this parameter, the software does not split vias.

Note: This parameter is obsolete and has been replaced by `-splitLongVia`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-splitLongVia {length spacing offset}`

Splits vias that are longer than the specified length into smaller vias that have the specified center-to-center spacing and bottom/left edge offset (all values in micrometers).

Note: This parameter replaces -split_long_via.

`-stopBlockPin {nearestTarget | boundary | boundaryWithPin | padRing | lastPadRing}`

Specifies the extension target for block pins.
Default: If you do not specify this parameter, block pins are extended to the nearest ring or stripe.

Specify one of the following:

`nearestTarget`

Extends the block pin to the nearest ring or stripe.

`boundary` Extends the block pin to the design boundary, without creating new power pins.

`boundaryWithPin`

Extends the block pin to the design boundary and creates a new power pin along the design boundary. Any overlaps with existing I/O pins at the design boundary are flagged as violations after the extension.

`padRing` Extends the block pin to the first pad ring that already exists on top of the I/O pad cells.

`lastPadRing` Extends the block pin to the last pad ring that already exists on top of the I/O pad cells.

Note: This parameter is obsolete and has been replaced by `-blockPinTarget {nearestRingStripe | nearestRing | nearestTarget | boundaryWithPin | padRing | farthestPadRing}`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-stopStripeSCPin {firstAfterRowEnd | boundary | boundaryWithPin | padRing | lastPadRing | none}`

Specifies the primary extension target for unconnected stripes and standard cell pins. If the selected target type is not available for a stripe, the stripe is left open. If the selected target type is not available for a standard cell pin, the target type specified for the `-secondaryStopSCPin` parameter is used. *Default:* If you do not specify this parameter, standard cell pins extend to the first ring or stripe outside of the row.

Select one of the following:

`firstAfterRowEnd`

Extends the stripe or standard cell pin to the first ring or stripe outside of the row.

`boundary` Extends the stripe or standard cell pin to the design boundary, without creating new power pins.

`boundaryWithPin`

Extends the stripe or standard cell pin to the design boundary and creates a new power pin along the design boundary. Any overlaps with existing I/O pins at the design boundary are flagged as violations after the extension.

`padRing` Extends the stripe or standard cell pin to the first pad ring that already exists on top of the I/O pad cells.

`lastPadRing` Extends the stripe or standard cell pin to the last pad ring that already exists on top of the I/O pad cells.

`none` Unconnected stripes and standard cell pins are not extended, and connections may be left open.

Note: This parameter is obsolete and has been replaced by `-stripeSCpinTarget`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-straightConnections [straightWithDRCClean] [straightWithChanges]`

Specifies that only straight connections are made between targets and that jogs are not allowed during routing.

Note: You must specify either the `-straightConnections` parameter or the `-jogControl` parameter.

If you specify the `-straightConnections` parameter, you can also specify one or both of the following options:

`straightWithDRCClean`

Leaves a route open if a straight connection cannot reach a target without causing a DRC violation. If you do not specify this option, the software makes a straight connection to a target, even if the route creates a DRC violations.

`straightWithChanges`

Permits the route to change to another layer to avoid DRC violations. If you do not specify this option, the software makes only straight connections on the same layer.

Note: This parameter is obsolete and has been replaced by `-allowJogging and -allowLayerChange`. This parameter works in this release, but will be removed in a future release. To ensure compatibility with future releases, remove this parameter from your scripts.

`-stripeJogViaMultiplier` *`viaSizeMultiplier`*

Specifies a real number to be used as a multiplier that increases the size of vias in stripes that need to jog and switch layers to connect to the target. For example, if you specify a value of `1.5`, the via will be one-and-a-half times the size of a full-size via. This can help reduce resistance.
Default: If you do not specify this parameter, the software uses a value of `1`.

`-stripeSCpinTarget {firstAfterRowEnd | boundaryWithPin | padRing | farthestPadRing | none}`

Specifies the primary extension target for unconnected stripes and standard cell pins. If the selected target type is not available for a stripe, the stripe is left open. If the selected target type is not available for a standard cell pin, the target type specified for the `-secondaryStopSCPin` parameter is used.
Default: If you do not specify this parameter, standard cell pins extend to the first ring or stripe outside of the row.

Note: This parameter replaces `-stopStripeSCpin`.

`-targetObjListFile` *`filename`*

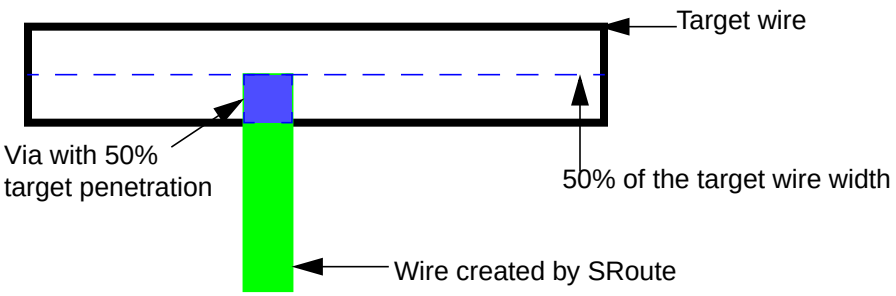
Specifies a `.obj` file that contains a list of objects that the sroute software is to either consider as a target or remove from the list of available targets.

Note: Cadence recommends that you use the Target List Editing page of the SRoute form to create this file. For more information, see SRoute in the “Route Menu” chapter of the *Encounter Menu Reference*.

`-targetPenetration` *`wiretype percent`*

Specifies a smaller size for vias connecting to targets of a particular wire or pin type.

By default, if a wire or pin is a target to which a via can be connected, the via is a full-sized. If you want the via to be smaller, you can specify a value in this field for each wire type. Only a single dimension of the via will be scaled. For example, if you specify a value of 50, the via will be full-sized in one dimension (the width of the wire created by SRoute), but only half the width in the other dimension (the width of the target wire).



Specify one or more of the following types of wires or pins, along with a percentage value:

- padring
- corering
- stripe
- blockring
- blockpin
- coverpin

`-targetViaBottomLayer` *`layerNumber`*

Specifies the lowest layer that can be used for via stacking at a target.
Default: If you do not specify this parameter, all layers are used.

`-targetViaTopLayer` *`layerNumber`*

Specifies the highest layer that can be used for via stacking at a target.
Default: If you do not specify this parameter, all layers are used.

`-verbose`

Specifies that each progress message appears in the log file and the Encounter console.

`-viaConnectToShape {blockring | stripe | ring | padring |blockwire | corewire | followpin | iowire}`

Specifies which shapes vias can connect to.
Default: If this parameter is not specified, vias can connect to all shapes.

`-viaThruToClosestRing`

Specifies that a via should connect only to the closest ring layer when multiple rings overlap.
Default: If you do not specify this parameter, vias connect to all overlapping rings.

Extra Configuration File Options for sroute

You should only use the extra configuration file if you are familiar with its use.

The following variables can be used in the extra configuration file:

- `srouteAlignViaOnStripe {true | false}`

  `This variable is obsolete and has been replaced by srouteSecondaryPinAlighViaOnStripe.`

- `srouteAutoRingPadBlockpinRoute {true | false}`

  Generates rings between pad pins and block pins. When this option is set to `true`, the Encounter software automatically generates the rings between pad pins and block pins to be used as targets for pad pins and block pins.

  The default is `false`.

- `srouteAutoRingLayer` *`layer_number`*

Specifies the layer of the rings.

- `srouteAutoRingOffsetFromBlockPin` *`db_unit`*

  Specifies the clearance between the rings and the block pins.

- `srouteConnectAreaIO [ring]`

  Connects area I/Os, and optionally, rings.

  This variable replaces `srouteConnectAreaIO` and `srouteConnectAreaIOAndRing`.

- `srouteConnectAreaIO {true | false}`

  Note: This variable is obsolete and has been replaced by `srouteConnectAreaIO [ring]`.

- `srouteConnectAreaIOAndRing {true | false}`

  Note: This variable is obsolete and has been replaced by `srouteConnectAreaIO [ring]`.

- `srouteConnectToCenterOfPin {true | false}`

  The default is `false`.

  Connects a wire to the center of the pad pin.

- `srouteConnectToEdgeOfBump {true | false}`

  The default is false.

  Routes an I/O pin to the edge of the bump.

- `srouteFcReportRes` *`fileName`*

  Generates a report summarizing the resistance constraints met by `fcroute`. You can identify any open nets from the report.

  Note: The `fcroute` estimated resistance report may not be as accurate as QRC extraction. You should always `runQRC` to get accurate RC values after `fcroute`.

- `srouteFollowPinOnShapePin {true | false}`

  The default is false.

  Specifies whether the primary pins are O-shaped.

- `srouteForbidStackVia {true | false}`

Note: This variable is obsolete and has been replaced by `srouteStairCaseVia`.

- `srouteGrouteOptimizeWidth {true | false}`

  The default is `false`.

  Note: You must use this option in conjunction with the `srouteGrouteOptimizeSpacing` option and they cannot be set to `true` at the same time.

  Automatically adjusts the width within the max and min constraint. When set to `true`, this configuration file option calls the global router to get the optimized width for each net and writes the results to a constraint file called `width.cons`.

  You can create your own constraint file to specify how the width is calculated. For example, if you have 10 nets in the design (`n1` through `n10`), you can create the following constraint file:

```
NETS
   MINWIDTH 2.0
   MAXWIDTH 5.0
   WIDTHSTEP 0.1
   n1 n2 n3
END NETS
NETS
   MINWIDTH 5.0
   MAXWIDTH 10.0
   WIDTHSTEP 0.5
   n4 n5 n6
END NETS
NETS
   WIDTH 3.0
   n7 n8
END NETS
WIDTH 12
```

  where:

`n1, n2, and n3`	Have a width between 2 and 5, and an incremental size of 0.1.
`n4, n5, and n6`	Have a width between 5 and 10, and an incremental size of 0.5.
`n7 and n8`	Have a fixed width of 3.0.

`n9 and n10` Have a fixed width of 12.0.

Note: All the nets within the same NET group will use the same final width. If you allow the nets to have different widths, while satisfying the min and max value, you can use the following configuration file option:

```
srouteGrouteUniformWidth false
```

- `srouteLayerNormalCost.`*`layerNumber costvalue`*

  `costvalue` is an integer in the range of `1` to `32`.

- `srouteLayerWrongWayCost.`*`layerNumber costvalue`*

  `costvalue` is an integer in the range of `1` to `32`.

- `srouteMinRailLength` *`minLength`*

  Limits the creation of followpin rails to a length at or above the specified value. SRoute will not create followpin rails with a length less than the specified value. You can use this variable to restrict the creation of followpin rails where row spacing is limited. Units in database units.

- `srouteRoutePowerBarPort {true | false}`

  The default is `true`.

  When set to `false`, SRoute does not attempt to route outside the block after the power bars are routed.

- `srouteSecondaryPinAlighViaOnStripe {true | false}`

  The default is `false`.

  Controls the alignment of stacked cross vias between followpin rails and stripes. When a followpin rail passes through a pair of stripes, the power routing software generates stacked vias to connect the followpin rail to the stripe of the same net. The stacked via is created between the stripes and spans from the followpin rail layer to

the layer just below the stripe layer. The via between the stripe layer and the layer below is placed at the intersection of the followpin rail and the stripe of the same net. The vias are connected with a wire from the layer just below the stripe layer.

The `srouteSecondaryPinAlighViaOnStripe` configuration file option is enabled when set to `true` and only when the following conditions are met:

- When you specify two nets with the sroute command. For example:

  ```
  sroute -nets {VDD GND}
  ```

- When stripes are of the same layer and of the same width
- When the distance between the stripes is less than their width
- When there are only two stripe wires within the distance of the width (not split)

Note: This variable replaces `srouteAlignViaOnStripe`.

- `srouteSecondaryPinIgnoreStandardCell {true | false}`

  In the previous release, if the followpins were on the `M1` or the `M2` layer, the command always treated the standard cells in the path of the followpin wires as blockages.

  In this release, you can set the option to `true` to specify that the standard cells in the path of the followpin wires should not be considered as blockages even if the followpins are on the `M1` or the `M2` layer. This way, the secondary pin routing can route through these standard cells.

  By default, the value is `false`. This means that the standard cells in the path of the followpin wires are treated as blockages if the followpins are on the `M1` or the `M2` layer.

- `srouteShifterPinObjListFile` *`file_name`*

  Specifies a file containing a list of secondary pins for sroute to ignore when routing. Use this variable when you have already routed certain shifter pins, if the shifter pins have different alignment, or if you want to use a different router (such as NanoRoute) to connect the pins, for example.

  The file format is as follows:

  `delete pin` *`pin_name`* `macro` *`macro_name`* `layer` *`layer_name`*

  In the file, you can repeat this line for any pins as you want.

  Currently, `layer` is ignored.

- `srouteStairCaseVia {true | false}`

The default is `false`.

Note: This variable replaces `srouteForbidStack`.

- `srouteUseSpecifiedWidthForTopLayer {true | false}`

  The default is `false`.

27

RC Extraction Commands

defineRCCorner

```
defineRCCorner
     {-late {best | typical | worst} [temp]
     -early {best | typical | worst} [temp]}
     | {-latespef fileName -earlyspef fileName}
```

Specifies the extraction values to be used during timing analysis. The extraction values can be for three process corners—best, typical, and worst.

Note: When the software is in multi-mode multi-corner analysis mode, the `defineRCCorner` command is disabled and cannot be used. Use the multi-mode multi-corner command set to configure the extraction environment.

Parameter

`best | typical | worst`
: Specifies the process corner for which the extraction values are required.

`-early`
: Specifies the extraction values to be used for hold analysis. Use this parameter in conjunction with the `-late` parameter.

`-earlyspef` *`fileName`*
: Specifies the SPEF file to be used for hold analysis. Use this parameter in conjunction with the `-latespef` parameter.

`-late`
: Specifies the extraction values to be used for setup analysis. Use this parameter in conjunction with the `-early` parameter.

`-latespef` *`fileName`*
: Specifies the SPEF file to be used for setup analysis. Use this parameter in conjunction with the `-earlyspef` parameter.

`temp`
: Specifies the operating temperature. The software derates the value of resistance based on this temperature. The unit of temperature is Celsius.
Default: The resistance value is not derated.

Example

The following command specifies the worst process corner to be used for setup and best to be used for hold analysis:

```
defineRCCorner -late {worst} 125 -early {best} 25
```

Related Topics

- timing.tcl in the *Encounter Flat Implementation Flow Guide*
- timing.tcl in the *Encounter Hierarchical Implementation Flow Guide*

extractRC

```
extractRC
     [-outfile fileName]
```

Extracts resistance and capacitance for the interconnects and outputs the results to a file. Use this command after performing trial routing or final routing, and after using the setExtractRCMode command.

Parameters

`-outfile fileName`	Specifies the name of the extracted capacitance file. **Note**: This parameter is only supported for default extraction and is not supported for multi-mode multi-corner analysis.

Example

The following command extracts interconnect resistance and capacitance values and writes the values to the `design_234.cap` file:

```
extractRC -outfile design_234.cap
```

Related Topics

- Signoff in the *Encounter Flat Implementation Flow Guide*
- Run the Partition Sign-Off Flow and Generate ILM Models in the *Encounter Hierarchical Implementation Flow Guide*
- Run the Top-Level Sign-Off Flow in the *Encounter Hierarchical Implementation Flow Guide*

generateCapTbl

```
generateCapTbl
     -ict fileName
     [-solverExe exec_path]
     [-lef fileName [-shrinkFactor value]]
     [-encrypt]
     [-incaptable fileName]
     [-cap totalCapFactor]
     [-xcap crossCouplingFactor]
     [-res resistanceFactor]
     -output fileName
     [-bestModel]
```

Reads in the interconnect technology (ICT) format file and generates a file containing a capacitance table.

The `generateCapTbl` command puts regular routing layers in the capacitance table file, and omits the poly layers local interconnect layers and mimcap layers. The `generateCapTbl` command recognizes poly layers by the `gate_forming_layer true` statement.

Note: The command accepts ICT files that contain the `gate_forming_layer true` construct in multiple `conductor` sections to allow handling of diffusion models as a conductor. In this case, the Encounter captable generation process ignores all but the last `gate_forming_layer true` construct. The last one is considered as `poly` and is used as the ground plate level for the metal layers above.

Important

Cadence recommends using the ICT file to generate the capacitance table.

The generated captable provides more simulation points and accounts for coupling capacitances to the wires on the layers above and below the victim wire, which are important considerations, especially for 65nm or below designs.

Important

You can use the multi-CPU processing commands to generate capacitance table in parallel mode. However, this functionality is not available for standalone capacitance table generation.

To generate a capacitance table file in parallel mode, use the following commands:

```
setDistributeHost -rsh -add { host1 host2 host3 }
setMultiCpuUsage -numHosts 3
generateCapTbl -ict sample.ict -output sample.capTbl
```

This functionality does not include multi-threading and super-threading operations. Therefore, the following options of the `setMultiCpuUsage` command are not supported:

- `-numThreads`
- `-superThreadsNumHosts`
- `-superThreadsNumThreads`

Parameters

`-bestModel` — With this parameter, the plates in the `+2`/`-2` layers of the extended capacitance table are removed. The capacitance results will be optimistic.

`-cap` *`totalCapFactor`*

Specifies the capacitance scaling factor. Use this parameter with the `-incaptable` parameter to generate a capacitance table with capacitance values scaled by the specified scaling factor.

`-encrypt` — Encrypts the generated capacitance table to protect the process information in the capacitance table header. When you use the `readCapTable` command to read the encrypted capacitance table, the software writes the version number of the Encounter software that you had used to generate the file, and the name of the process to the log file.

`-ict` *`fileName`*

Specifies the name and path to the ICT (interconnect technology) format file. Manufacturing effects, such as general bias (etch) or etch table (with dependancy on width and spacing) are all taken from the ICT file. This information is provided to the captable and can be found in the header. For example, the top and bottom enlargement is used for general etch to describe the difference between the drawn width (if after scaling with `shrinkfactor`) and the final average width on silicon.

Taking all of the manufacturing information from the ICT file, which is also used as source for the QRC signoff extraction technology file, also ensures a good match with the signoff extractor.

To support multiple process corners, a capacitance table should be created for each corner from the matching corner ICT file.

Note: All resistivity information is taken from the captable and not from the LEF file.

`-incaptable` *`fileName`*

Specifies the name of an existing capacitance table in ASCII format. You use this parameter along with the `-encrypt` parameter to create an encrypted version of an existing capacitance file.

`-lef` *`fileName`*

Specifies the name and the full path to the LEF file. Use this parameter to generate an extended capacitance table with precise non-default rule specification and non-default width and spacing defined in the LEF file. With the LEF file, this parameter generates a capacitance table by including the non-default rules defined in the LEF. Without the LEF, the software uses generic widths and spacings, which might increase the need for interpolation during extraction.

`-output` *`fileName`*

Specifies the output filename that contains the capacitance table.

`-res` *`resistanceFactor`*

Specifies the resistance scaling factor. Use this parameter with the `-incaptable` parameter to generate a capacitance table with resistance values scaled by the specified scaling factor.

`-shrinkFactor` *`value`*

Specifies the value by which the measurements in the optional LEF file have to be shrunk for the technology used with this design. If you use this parameter with the `-lef` parameter and the LEF technology does not match the target technology in the ICT file, Encounter scales the dimensions when reading the LEF file. The ICT file should always be used for the target technology (min width and min spacing).

Note: When working on a design that needs to be scaled down in Encounter, you must also set the shrinkfactor in *Design Import*. This tells the extractor to scale down all geometries before processing.

When the ICT file contains a layout_scale in the process declaration part, you do not need to specify the shrink factor. In this case, the embedded shrink factor will also be written in the captable header. And, there is no need to set the shrink factor in the Design Import form. The embedded layout_scale is used to support half nodes such as 40nm technology.

Default: `1.0`

`-solverExe` *`solverexe`*

Specifies the optional full path name to the field solver executable (QuickCap or Coyote). For example:

Coyote:
`/cad/SOC32/sun4v/tools/bin/coyote`

Note: By default, the software uses Coyote field solver executable installed in the Encounter hierarchy, so you are not required to specify this path.

`-xcap` *`crossCouplingFactor`*

Specifies the cross-coupling capacitance scaling factor. Use this parameter with the `-incaptable` parameter to generate a capacitance table with cross-coupling capacitance values scaled by the specified scaling factor.

Example

The following command reads in the ICT file `test.ict`, and generates a `new.capTbl` output file:

```
generateCapTbl -ict test.ict -output new.CapTbl
```

generateRCFactor

```
generateRCFactor
     [-all]
     [-spefIn spefInFile]
     [-defaultRC | -detailRC]
     [-trialRoute | -nanoRoute]
     [-noCheckLefRes]
     [-useOstrich]
```

Generates the capacitance scaling factor by comparing the SPEF files generated by native extraction and sign-off extraction. After generating the scaling factors, the command builds the timing graph. If the correlation of the worst negative slack improves with the scaling factor, the scaling factor is set, otherwise it is reported as the suggested scaling factor.

Note: Currently, this command does not support MMMC designs.

This command runs native default or detailed extraction depending on the parameter that you specify. You can either specify a sign-off file for comparison using the `-spefIn` parameter or the software runs QRC sign-off extraction to generate a sign-off SPEF. You must load the QRC technology file for the command to run the QRC extraction. If the design is not placed or routed, you can specify the routing parameters, `-trialRoute` or `-nanoRoute`, for the command to place and route the design before generating the scaling factor.

Note: A QRC standalone extraction licence is required to run sign-off extraction in this command. If you do not have a QRC standalone extraction license, you can use the `-spefIn` parameter to specify the sign-off SPEF file(s).

Parameters

`-all`	Compares both native default extraction and native detailed extraction SPEF with sign-off SPEF or SPEF file that you specify using `-spefIn` parameter. The comparison of native default extraction results in default capacitance scaling factor. The comparison of native detailed extraction results in detail and coupling capacitance scaling factor. When you use `-all` to compare with sign-off SPEF using QRC, if your design was not routed earlier, by default the software performs trialRoute before comparing native default extraction, and runs the NanoRoute router before performing native detailed extraction. When you use `-all` to compare with SPEF file that you specify using the `-spefIn` parameter, the design must be placed, detail routed and the SPEF file generated using a sign-off extraction tool before using the `generateRCFactor` command. You must use the same routing for both native and sign-off SPEF files.
`-defaultRC`	Compares the native default extraction SPEF with the QRC sign-off SPEF or SPEF file that you specify using the `-spefIn` parameter. When you use the `-defaultRC` parameter, the software performs trialRoute by default if your design was not routed earlier. You can use the `-nanoRoute` parameter to run the NanoRoute router instead of trialRoute.
`-detailRC`	Compares the native detailed extraction SPEF with the QRC sign-off SPEF or SPEF file that you specify using the `-spefIn` parameter. When you use the `-detailRC` parameter, the software runs the NanoRoute router by default if your design was not routed earlier. You can use the `-trialRoute` parameter to perform trialRoute instead of running the NanoRoute router.
`-nanoRoute`	Runs the NanoRoute router if the design was not routed earlier.
`-noCheckLefRes`	Specifies not to check the resistivity of every layer in the LEF file with the QRC technology file. *Default*: Checks the resistivity of every layers in the lef file with the QRC technology file and fixes the mismatch by creating a new LEF file.

`-spefIn` *`spefInFile`*	Specifies the sign-off SPEF file to be used for comparison. When you use this parameter, the design must be placed, detail routed and the SPEF file generated using a sign-off extraction tool before using the `generateRCFactor` command. You must use the same routing for both native and sign-off SPEF files.
`-trialRoute`	Performs trialRoute if the design was not routed earlier.
`-useOstrich`	Use Ostrich to compare the SPEF files generated by native extraction and sign-off extraction. *Default:* Uses the `spefCapCmp` perl script to compare the SPEF files.

Examples

- The following command generates the capacitance scaling factor by comparing the SPEFs generated by native extraction (both default and detailed) and sign-off extraction:

  ```
  generateRCFactor -all
  ```

  In this example, the design might not be placed or routed. By default the software places the design, and performs trialRoute before comparing native default extraction, and runs the NanoRoute router before performing native detailed extraction.

  Following are some sample sections in the log file for the above command:

  In Detail RC Extraction Mode Using the NanoRoute Router

  ```
  ============================================================
  The suggested capacitance scale factor is              0.9371
  The suggested coupling capacitance scale factor is     1.5478
  ============================================================
  ============================================================
  With FE detail extraction before correction     WNS = -12.077
  With FE detail extraction after  correction     WNS = -11.240
  With reference RC data                          WNS = -10.386
  ============================================================
  ============================================================
  The corrleation between FE extraction and reference RC data is improved.
  The capacitance scale factor is set to               0.9371
  The coupling capacitance scale factor is set to      1.5478
  ============================================================
  ```

In Default RC Extraction Mode Using TrialRoute

```
============================================================
The suggested default capacitance scale factor is         1.194
============================================================
============================================================
With FE default extraction before correction    WNS = -7.985
With FE default extraction after  correction    WNS = -10.905
With reference RC data                          WNS = -10.386
============================================================
============================================================
The corrleation between FE extraction and reference RC data is improved.
The default capacitance scale factor is set to       1.194
============================================================
```

- The following command generates the capacitance scaling factor by comparing the SPEF generated by native extraction (both default and detailed) and sign-off SPEF specified using the `-spefIn` parameter:

  ```
  generateRCFactor -all -spefIn test.spef
  ```

  In this example, the SPEF file specified using the `-spefIn` parameter must be generated using a sign-off extraction tool on a placed and detail routed design. You must use the same routing for both native and sign-off SPEF files.

getExtractRCMode

```
getExtractRCMode
     [-assumeMetFill]
     [-capFilterMode]
     [-compressOptMemRCDB]
     [-coupled]
     [-coupling_c_th]
     [-defViaCap]
     [-effortLevel]
     [-engine]
     [-extraCmdFile]
     [-force]
     [-hardBlockObs]
     [-incremental]
     [-ipdb]
     [-noGroundXCap]
     [-noReduce]
     [-optMemory]
     [-quiet]
     [-rcdb]
     [-reduce]
     [-relative_c_th]
     [-scOpTemp]
     [-specialNet]
     [-total_c_th]
     [-useLEFCap]
     [-useLEFResistance]
     [-useNDRForClockNets]
     [-useShieldingInDetailMode]
     [-viaCap]
```

Displays the following information about a setExtractRCMode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setExtractRCMode` parameters.

Parameters

parameter_names	Displays information for the specified parameters. You can specify one or more parameters. See setExtractRCMode for descriptions of the clock tree synthesis parameters you can specify.
-quiet	Displays the current settings for the specified parameters in Tcl list format only. If you specify -quiet without any parameters, the software displays the current settings of all setExtractRCMode parameters in Tcl list format.

Example

- The following command displays the current setting for the -viaCap parameter:

  ```
  getExtractRCMode -viaCap
  ```

  The software displays the following information:

  ```
  -viaCap true                        # bool, default=true
  true
  ```

- The following command displays the current settings for the -viaCap and -engine parameters:

  ```
  getExtractRCMode -viaCap -engine
  ```

  The software displays the following information:

  ```
  -viaCap true                        # bool, default=true
  -engine default                  # enums={default detail CCE}, default=default
  {viaCap true} {engine default}
  ```

- The following command displays the current setting for the -viaCap parameter in Tcl list format only:

  ```
  getExtractRCMode -viaCap -quiet
  ```

  The software displays the following information:

  ```
  true
  ```

- The following command displays the current settings for the -viaCap and -engine parameters in Tcl list format only:

  ```
  getExtractRCMode -viaCap -engine -quiet
  ```

  The software displays the following information:

```
{viaCap true} {engine default}
```

- The following command displays the current settings for all `setExtractRCMode` parameters in Tcl list format only:

  ```
  getExtractRCMode -quiet
  ```

getQRCTechfile

```
getQRCTechfile
     [-opTemp | -layermap | -techfile]
     [-quiet]
```

Returns the name of the QRC technology file, operating temperature value, and the location of the layer map file for CCE extraction. You can set these values using the setQRCTechfile command.

Parameters

`-layermap`	Returns the location of the optional layer map file.
`-opTemp`	Returns the operating temperature value.

`-quiet`	Returns an abbreviated output that only shows the names or current settings of the parameters set with the `setQRCTechfile` command. For example:
	
	`getQRCTechfile`
	`#techfile 'simple.tech' (string, user setting)`
	`#layermap ''(string, not specified)`
	`#optemp 25 (float, default setting)`
	`{techfile simple.tech}{layermap}{25}`
	
	`getQRCTechfile -quiet`
	`{techfile simple.tech}{layermap}{25}`

`techfile`	Returns the name of the QRC technology file.

Example

- The following command specifies the name of the QRC technology file:

```
getQRCTechfile -techfile
#techfile 'simple.tech' (string, user setting)
simple.tech
```

rcOut

```
rcOut
     {-instance hierInstanceName -setportload fileName
     | -setload fileName
          [-excNetFile fileName | -net fileName]
          [-best | -typical | -worst]
     | -setres fileName
          [-excNetFile fileName | -net fileName]
          [-best | -worst | -typical]
     | -spef fileName [-unmapped]
               [-excNetFile fileName | -net fileName [-addHeaderTail]]
               [-best | -worst | -typical]
     | -spf fileName [-excNetFile fileName | -net fileName [-addHeaderTail]]
               [-filesize sizeInMbytes]
     | -verilog fileName}
     [-cUnit unit]
     [-opTemp temp]
     [-view viewName | -rc_corner rcCornerName]
     [-noRes]
```

Reads the parasitic database and outputs the parasitics in one of the following formats:

- setload
- setres
- Standard Parasitic Exchange Format (SPEF)
- Standard Parasitic Format (SPF)
- Verilog
- setportload

Parameters

`-best | -typical | -worst`

Specifies the process corner for which the extraction values are required.

Note: These parameters work only with the detailed and default extraction modes. They are not supported in CCE extraction mode or when using the multi-mode multi-corner (MMMC) flow.

Default: `-typical`

`-addHeaderTail` Adds the header and footer information in the output SPF or SPEF file. This parameter is only available with the `-net` parameter.

`-cUnit` *`unit`* Specifies the unit of capacitance as `fF` or `pF` in the output SPEF file.

Default: `pF`

`-excNetFile` *`fileName`*

Specifies the name of the file that contains nets to be excluded from the output file.

`-filesize` *`sizeInMbytes`*

Specifies the size of the output SPF file. When the software reaches this size limit, it writes the extracted data to multiple files. The output file includes all the SPF files using the `.include` command.

`-instance` *`hierInstanceName`*

Specifies the hierarchical instance to be extracted for its port loading of external loads for `setportload` format. The external loads are wire loading, internal pin loading, and external pin loading.

`-net` *`netName`* Specifies the name of the input file that contains the net names to be extracted.
Default: Extracts all net names

`-noRes` Generates a SPEF file without the resistance information. The SPEF file generated by this parameter can only be used to improve the performance of third-party simulator tools in early stages of the design.

Caution: Most of the tools expect the resistance information to be present in the SPEF file for performing extraction.

`-opTemp` *`temp`* Specifies the operating temperature. The software derates the value of resistance based on this temperature. The unit of temperature is Celsius.
Default: The resistance value is not derated.

Note: This parameter is not supported in multi-corner or CCE extraction modes.

`-rc_corner` *`rcCornerName`*

	Specifies the RC corner that is used to generate the output SPEF file in the multi-corner flow.
`-setload` *`fileName`*	Specifies the name of the file in which the value of lump sum capacitance on nets is stored.
`-setportload` *`fileName`*	
	Specifies the name of the output file. All output formats are written to this file. **Note:** This option is only available in the default extraction mode.
`-setres` *`fileName`*	Specifies the name of the file in which the resistance value on nets is stored.
`-spef` *`fileName`*	Specifies the name of the SPEF output file.
`-spf` *`fileName`*	Specifies the name of the SPF output file.
`-unmapped`	Specifies the exclusion of name mapping in the output SPEF file. This parameter is only available in the default mode.
`-verilog` *`fileName`*	Specifies the name of the output Verilog file.
`-view` *`viewName`*	Specifies that the RC corner associated with the specified view is used to generate the output SPEF file in the multi-corner flow.

Examples

- Reads the parasitic database created during RC extraction and writes the interconnect capacitance in the setload format to the file `TOPCHIP_SP.setload`:

  ```
  rcOut -setload TOPCHIP_SP.setload
  ```

- Reads the parasitic database created during RC extraction and writes the port loading of instance TOP/MAC to the specified file:

  ```
  rcOut -instance TOP/MAC -setportload TOP_MAC.setportload
  ```

Related Topics

- Signoff in the *Encounter Flat Implementation Flow Guide*
- Run the Partition Sign-Off Flow and Generate ILM Models in the *Encounter Hierarchical Implementation Flow Guide*
- Run the Top-Level Sign-Off Flow in the *Encounter Hierarchical Implementation Flow Guide*

readCapTable

```
readCapTable
     {-typical fileName1 -best fileName2 -worst fileName3 | fileName}
```

Reads the capacitance table file(s) used for extracting interconnect capacitance values. You can use this command to read the typical, best, and worst capacitance table or read only one table for a specific corner.

Note:

- You must load the design LEF file(s) into the Encounter database before executing this command.
- When the software is in multi-mode multi-corner analysis mode, the `readCapTable` command is disabled and cannot be used. Use the `create rc corner` command for multi-mode multi-corner analysis.

Parameters

fileName	Specifies the name of the input file.
`-typical` *fileName1* `-best` *fileName2* `-worst` *fileName3*	Specifies the name of the input files for typical, best and worst cases. **Note:** You must specify at least the best and the worst case input files when reading the capacitance table information using this parameter set.

Examples

- The following command reads the existing capacitance table file `t18.capTbl`:

  ```
  readCapTable t18.capTbl
  ```

- The following command reads the capacitance table files, `cap1.tb1`, `cap2.tb2`, and `cap3.tb3` corresponding to typical, best, and worst case respectively:

  ```
  readCapTable -typical cap1.tb1 -best cap2.tb2 -worst cap3.tb3
  ```

reportCapTable

`reportCapTable`

Reports the current capacitance values in the capacitance table file (for basic capacitance table only). Use this command after generating or loading the capacitance table.

Parameters

None

restoreRC

`restoreRC` *`path_to_rcdb_dir`*

Restores the RC extraction data that was previously saved with the `saveDesign` *`name`* `-rc` command.

Note: The systems you use to save and restore RC extraction data must match (32-bit or 64-bit data models).

Parameters

`path_to_rcdb_dir`

Specifies the location of the RCDB directory.

runLibGen

```
runLibGen
     {-cmd filename
     | [-inputType {LEF | GDS | LEF&GDS}]
     [-techFile fileName]
     [-libraryName lib_name]
     [-layerMapping fileName]
     -lefFileList fileName
     [-gdsFileList fileName]
     [-cellList cellList]
     [-genPortPowerView]
     [-powerPinList {Pinname voltage}+]
     [-groundPinList {pinName voltage}+]
     [-genDetailedPowerView]
     [-defaultFrequency value]
     [-xtcCommandFile fileName]
     [-fireCommandFile fileName]
     [-thunderModelDir dirName]
     [-noCaseSensitivity]
     [-copyPortToGray]
     [-copyPortToObs]
     [-haltOnUndefinedLefLayer]
     [-verbose]
     [-maxerrors value]}
     [-detailedPowerGateFile fileName]
     [-portPowerGateFile fileName]
```

Takes LEF, GDSII, or a combination of LEF and GDSII data as input and creates a binary view of the cell contents within the LibGen cell library database. The binary views represent the graycell data or the power grid views (or both) for each cell included in the library. The default name of the file generated by `runLibGen` is `library.cl`. The generated library can be used by QRC standalone sign-off extraction and the VoltageStorm™ power grid verification solution.

Note: QRC standalone signoff extraction now can read LEF files directly. The library creation step through runLibgen is no longer needed to run QRC. The recommended flow is to use direct LEF input. Libgen executable used by the runLibgen command can only be found in the hierarchy provided with the Voltagestorm product.

Parameters

`-cellList` *`cellList`* Specifies the cells that LibGen should process. Use the * character to represent all cells. If you do not specify a value for this parameter, LibGen processes all cells.

`-cmd` *`fileName`*
: Specifies the command filename that contains predefined values. Use this file to save and use data for later runs. For more information, see the examples for a sample command file. *Default: `libGen`*`.cmd`

`-copyPortToGray`
: Copies the port and feedthrough geometries into the graybox data used for timing extraction.

`-copyPortToObs`
: Copies the port and feedthrough geometries into the obstruction view, which you can use for timing extraction.

`-defaultFrequency` *`value`*

: Specifies the default operating frequency for blocks. Use this parameter to generate detailed power grid views for power analysis, using the Accura technology. For more information, see the LibGen chapter in the *Data Preparation Manual* for Cadence power analysis grid and extraction tools.

`-detailedPowerGateFile` *`fileName`*

: Specifies the name of the detailed power gate file that VoltageStorm uses to perform power gate steady-state analysis. You can generate the detailed power gate file using the following VoltageStorm `detailed_power_gate` command syntax:

```
detailed_power_gate CELL cellname
SUPPLY supply_net_name SWITCHED switched_pin_name
```

where:

- `CELL` *`cellname`* specifies the name of the cell.
- `SUPPLY` *`supply_net_name`* specifies the name of the supply net
- `SWITCHED` *`switched_pin_name`* specifies the name of the switched power pin.

Note: If you specify the `RON`, `IDSAT`, and `ILEAK` options, the data will override the data that is extracted by Thunder.

For more information, see the *LibGen: Creating Cell Libraries* chapter of the *VoltageStorm Data Preparation Manual*.

`-fireCommandFile` *`fileName`*

Specifies the name of the Fire extraction command file. This parameter is required for power analysis. For more information on the command file, see the *VoltageStorm Data Preparation* manual.

`-genDetailedPowerView`

Generates a detailed power view. A detailed power view is a detailed electrical model of an internal power routing of a cell or macro, which is extracted from GDS data. The electrical model is contained in a detailed power grid. It provides resistance and current loading information to VoltageStorm. You use this parameter for blocks in the design that you know will affect the power consumption of the top-level design.

`-genPortPowerView` Generates a power grid library with port views. The port view contains power port information that provides the current loading to the power grid networks in VoltageStorm. Using this view is similar to running at blackbox mode. Typically you use this view for standard cells or blocks that will not affect power consumption of your circuit.

`-gdsFileList` *`fileName`*

Specifies the GDS file(s). Separate multiple file names with spaces.

`-groundPinList {`*`pinName voltage`*`}+`

Specifies ground pin names along with corresponding operating voltages.

`-haltOnUndefinedLefLayer`

Halts processing if LibGen encounters a LEF layer that was not defined in the layer map in the command file.

`-inputType {LEF | GDS | LEF&GDS}`

Specifies the input type as LEF, GDS, or both LEF and GDS.

`-layerMapping` *`fileName`*

Maps the layer name used in the LibGen command file to either the associated layer's GDSII layer and data type numbers, or to the associated layer's LEF or DEF layer name. For more information, see the examples for a sample of the layer mapping file.

`-lefFileList` *`fileName`*

Specifies the LEF file(s). Separate multiple file names with spaces.

`-libraryName` *`lib_name`*

Specifies the name of the output cell library file.
Default: `library`

`-maxerrors` *`value`*

Specifies the maximum number of errors to be reported before exiting out of the utility.
Type: Integer
Range: `1-10000`
Default: `1000`

`-noCaseSensitivity`

Specifies whether LibGen interprets the data in the LEF file as case-insensitive. If you do not specify this parameter, LibGen interprets the data in the LEF file as case-sensitive.

`-portPowerGateFile` *`fileName`*

Specifies the name of the port power gate file that VoltageStorm uses to perform power gate steady-state analysis. You can generate the port power gate file using the following VoltageStorm `port_power_gate` command syntax:

```
port_power_gate CELL cellname
SUPPLY unswitched_pin_name SWITCHED switched_pin_name
RON r_value IDSAT idsat_value ILEAK ileak_value
```

where:

- `CELL` *`cellname`* specifies the name of the cell.
- `SUPPLY` *`unswitched_pin_name`* specifies the name of the unswitched power pin. This is the pin that the leakage current is attached to if the power gate is in the off state
- `SWITCHED` *`switched_pin_name`* specifies the name of the switched power pin.
- `RON` *`r_value`* specifies the on-resistance value in ohms.
- `IDSAT` *`idsat_value`* specifies the value of the saturation current in milliamps.
- `ILEAK` *`ileak_value`* specifies the leakage current in milliamps.

For more information, see the *Power Gating* chapter of the *VoltageStorm Cell-Level Rail Analysis User Guide*.

`-powerPinList {`*`Pinname voltage`*`}+`

Specifies power pin names along with corresponding operating voltages.

`-thunderModelDir` *`dirName`*

Specifies the location of the Thunder model tables that are used to generate the current data for the cells in the library.

`-techFile` *`fileName`* Specifies the QRC technology file. The technology file contains interconnect models that QRC uses to extract the interconnect parasitic capacitance and resistance.

`-verbose` Controls the amount of information written to the LibGen log file. When you use this parameter, the software writes out detailed run-time messages in the log file.

`-xtcCommandFile` *`fileName`*

Specifies the name of an XTC file, which contains information about how devices are formed and how they are connected.

Example

- The following command creates a binary view of the cell contents within the LibGen cell library database with LEF file `tsmc13_6lm_tpz013g3.lef`, layer mapping file `lefdef_layer.map.txt`, and QRC technology file `qrc.tch` as inputs:

  ```
  runLibGen -copyPortToGray -copyPortToObs -verbose
     -lefFileList { tsmc13_6lm_tpz013g3.lef } -layerMapping lefdef_layer.map.txt
     -techFile qrc.tch -libraryName library_new
  ```

  The command copies the port and feedthrough geometries into the graybox and obstruction views.The name of the output cell library file is `library_new`.

- The following commands show the format of a typical command file:

  ```
  input_type pr_lef
  lef_file_list { tsmc90nm_55_6ml_mem.lef }
  cell_list { * }
  setvar library_name library
  setvar tech_file qrc.tch
  include lefdef.layermap
  setvar defgds_mbb_layername DEF_MBB
  ```

- The following commands show the format of a typical layer mapping file:

```
poly  POLYCIDE lefdef Poly
metal METAL_1  lefdef Metal1
metal METAL_2  lefdef Metal2
metal METAL_3  lefdef Metal3
metal METAL_4  lefdef Metal4
metal METAL_5  lefdef Metal
via   VIA_1    lefdef Via12
via   VIA_2    lefdef Via23
via   VIA_3    lefdef Via34
via   VIA_4    lefdef  Via45
```

runQRC

```
runQRC
     [-cmd fileName]
     [-multiCpu value]
     [-noSpecialNet]
     [-noRegularNet]
     [-includeFile file_list | -excludeFile file_list]
     [-includePattern pattern_list | -excludePattern pattern_list]
     [-rcType {decoupledRc | coupledRc}]
     [-techFile tech_file ]
     [-outputFileName fileName]
     [-compressedOutput]
     [-busChar value]
     [-pinChar value]
     [-hierarchyChar value]
     [-lefcapOutput]
     [-logFile fileName]
     [-maxErrorMessages value]
     [-maxResistorLength value]
     [-messageDetailLevel value]
     [-outputFileMaxSize value]
     [-noReduceOutput]
     [-couplingCThreshold value]
     [-relativeCThreshold value]
     [-totalCThreshold value]
     [-outputIncompleteNets]
     [-netCcOutput]
     [-outputUnconnectedPins]
     [-layoutScale value]
     [-promotePinPad {logical | none}]
     [-extraConfig fileName]
     [-spefOutput [extended | generic | PrimeTime | starN | CeltIC | none]]
     [-lsfCommand cmd]
     [-lsfNumber num]
     [-multiMachine host]
     [-lefFileList lefFileList -layerMapping layerMappingFile
     [-grayData {obs | none}] | -libraryName libraryName -grayData {gds | none}]
```

Performs QRC standalone sign-off RC extraction and upon completion, reads the resulting SPEF into the Encounter database. The Encounter software provides sign-off extraction functionality using QRC extraction. To use the `runQRC` command, specify the path to the QRC installation before running Encounter.

When you use the `runQRC` command in multi-mode multi-corner analysis, the software runs QRC standalone sign-off RC extraction for all active rc corners.

You can also use the Encounter multi-CPU processing commands to run the `runQRC` command. Use the following set of commands to run the runQRC command in distributed processing mode:

```
setDistributeHost -rsh -add { host1 host2 host3 }
setMultiCpuUsage -numHosts 3
runQRC
```

However, this functionality does not include multi-threading and super-threading operations. Therefore, the following options of the `setMultiCpuUsage` command are not supported:

- `-numThreads`
- `-superThreadsNumHosts`
- `-superThreadsNumThreads`

Note: If you wish to run the `runQRC` command on multiple processors on the same machine in which you are running Encounter, use the following commands:

```
setDistributeHost -local
setMultiCpuUsgae -numHosts 3
runQRC
```

The above set of commands process the `runQRC` command on three processors of the local machine.

Parameters

`-busChar` *`value`*
: Specifies the bus notation.

  Type: String

`-cmd` *`fileName`*

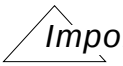
Specifies the command filename that contains predefined values. Use this file to save and use data for later runs.

The `-cmd` parameter must be run separate from other parameters of the `runQRC` command except the `-multiCPU`, `-multiMachine`, `-lsfNumber`, `-lsfCommand`, and `-extraConfig`. You can use the `-extraConfig` parameter to add a limited number of extraction directives (in qc style format) in an extra file that will be appended to the main command file.

For more information on variables that you can specify, see the *QRC Gate-Level Extraction* manual.

For a sample command file, see Sample Command File on page 1185.

Default: `qx.cmd`



Option `-cmd` is obsolete starting in the 8.1 release.

`-compressedOutput`

Enables QRC to write out a compressed spef using gzip.

`-couplingCThreshold` *`value`*

Specifies the threshold value that determines when the extractor lumps a net's coupling capacitance to ground. This parameter is applicable only when the `-coupled` parameter is set to `true`.

The software decouples the coupling capacitance of nets when the total coupling capacitance between the pair of nets is lower than the threshold specified with this parameter. The actual decoupling also depends on the setting of the `output_cap_filtering_mode` QRC variable, which can be specified in the QRC command file.

Default: `3 fF`. If the `setDesignMode -process` *`process_node`* command is used, the software assigns the recommended default value for this parameter based on the process node specified.

Tip

See the `setDesignMode` command for the list of default values that are applied based on the process node setting.

`-excludeFile` *`file_list`*

Specifies that the nets listed in the *`file_list`* must be excluded from extraction.

`-excludePattern` *`pattern_list`*

Specifies that the nets that match the patterns listed in the *`pattern_list`* must be excluded from extraction.

`-extraConfig` *`fileName`*

Specifies the name of the file that contains additional commands for QRC extraction. For more information on additional options that you can specify, see the *QRC Gate-Level Extraction* manual.

`-grayData {gds | obs | none}`

Performs a graybox capacitance extraction, which recognizes the effects of the cells on the interconnect.
Default: `gds`
Choose one of the following:

`gds`	Specifies that the source of the gray data is the GDS data type from the GDS cell library.
`obs`	Specifies that the source of the gray data is the obstruction data type from the LEF cell library.
`none`	Specifies that the Encounter software performs blackbox extraction rather than graybox extraction. Blackbox extraction ignores the effects of the cells on the interconnect.

`-hierarchyChar` *`value`*

Specifies the hierarchy notation.
Type: String

`-includeFile` *`file_list`*

Specifies that the nets listed in the *`file_list`* must be included in extraction.

`-includePattern` *`pattern_list`*

Selects all nets with names that contain the pattern(s) specified in the *`pattern_list`*.

`-layerMapping` *`layerMappingFile`*

Specifies the layer mapping file used to map the LEF data to the QRC technology file. Use this parameter in conjunction with the `-lefFileList` parameter to support the direct LEF flow.

Note: If you do not use the direct LEF flow, you must use the `-libraryName` parameter to specify the cell library file.

For QRC standalone extraction, Encounter uses the same layer mapping file syntax that was used for QX. The Encounter interface writes a command file for QRC in the QX-compatible language. If you want to write an extra configuration file, you should use the QX-compatible language. For more information, see the *Fire & Ice QXC to QRC Migration Guide*.

`-layoutScale` *`value`*	Specifies the scaling value by which the dimensions in the layout need to be scaled. The value specified with this option should match the shrink factor used in Encounter. (QRC can process designs that have been sized by a scale factor) *Default:* `1` (no scaling) *Type:* Real *Range:* `0.01` to `5`
`-lefcapOutput`	Outputs area capacitance per area and peripheral capacitance per edge in LEF format, for use by physical design tools.

`-lefFileList` *`lefFileList`*

Specifies the LEF files that contain the obstruction data for standard cells, I/Os, and macros. You must use this parameter in conjunction with the `-layerMapping` parameter to map the LEF data to the QRC technology file.

Note: The `-grayData gds` argument is not supported with the direct LEF flow. If you do not use the direct LEF flow, you must use the `-libraryName` parameter to specify the cell library file.

`-libraryName` *`libraryName`*

Specifies the name of the cell library file. You can either specify an existing library filename, or use the `runLibGen` command to generate a library file.

Note: The `-grayData obs` argument is not supported with this parameter.

`-logFile` *`fileName`* Specifies the name of the log file that the extractor generates.

`-lsfCommand` *`cmd`*

Supports distributed processing mode. Specifies the LSF, Grid Matrix System, or SSH protocol command that will be used to assign extraction jobs to available remote host machines. The protocol command assigns extraction jobs to a specified number (`-lsfNumber`) of machines over the network. Machines are chosen based on their current work load and availability.

Note: If you know the name of the remote host machine(s), you can use the `-multiMachine` parameter instead.

The `bsub` command is used for the LSF protocol and the `gridsub` command is used for the Grid Matrix System protocol.

Default: "`bsub -q normal -K`" (LSF)

For example:

```
runQRC -lsfNumber 4 -lsfCommand "bsub -q normal -K"
-cmd block.cmd block.def

runQRC -lsfNumber 8 -lsfCommand 'gridsub -wait -R
"OSNAME==Linux && SFIARCH==OPT64" '-cmd block.cmd
block.def
```

Note: You must use this parameter in conjunction with the `-lsfNumber` and `-cmd` parameters. Do not use this parameter with the `-multiMachine` parameter.

Option `-lsfCommand` is obsolete and has been replaced by Encounter distributed processing commands.

`-lsfNumber` *`num`*

Supports distributed processing mode. Specifies the number of remote host machines that will be used to run the extraction jobs.

Default: 4 (maximum of 12)

Note: You must use this parameter in conjunction with the `-lsfCommand` and `-cmd` parameters. Do not use this parameter with the `-multiMachine` parameter.

Option `-lsfNumber` is obsolete and has been replaced by Encounter distributed processing commands.

`-maxErrorMessages` *`value`*

Sets the maximum number of error messages issued by the extractor during a session.

Default: `100`

Type: Integer

`-maxResistorLength` *`value`*

Sets the maximum length for a resistor. Smaller values increase the accuracy of the distributed RC database, decrease capacity, and increase run times.

Default: `800`

Type: Integer

Range: `10` to `800` µm

`-messageDetailLevel` *`value`*

Sets the maximum number of warning messages issued by the extractor during a session.

Default: `10`

Type: Integer

`-multiCpu` *`value`*

Selects multiprocessor mode and specifies the number of processors to be used ranging from 2 to 4. Supports local machines with multiple processors.

Default: 2

Important

Option `-multiCpu` is obsolete. To specify the number of processors to use on a single machine, use the following commands:

```
setDistributeHost -local
setMultiCpuUsage -numHosts 3
```

`-multiMachine` *`host`*

Supports distributed processing mode. Specifies the name of the remote host machine that will be used to run the extraction job. To specify multiple host machines, you can enter multiple entries. Use this parameter if you know the name of the remote host machine (s). If you do not know the name of the remote host machine(s), you can use the LSF, Grid Matrix System, or SSH protocol (`-lsfCommand` and `-lsfNumber`) to have the extraction job assigned to an available remote host machine.

Note: You must use this parameter in conjunction with the `-cmd` parameter. Do not use this parameter with the `-lsfCommand` and `-lsfNumber` parameters.

Important

Option `-multiMachine` is obsolete and has been replaced by the Encounter distributed processing commands.

`-netCcOutput`

Writes out the following types of capacitance:

- The lumped coupling capacitance between two nets (net-to-net coupling capacitance).
- The total capacitance of each net.
- The total capacitance of two nets plus the total coupling capacitance between them.

`-noReduceOutput`

Specifies that the RC network is not reduced in the output.

`-noRegularNet`

Excludes from extraction all nets that are not special or mixed.

`-noSpecialNet`

Specifies that special nets should not be extracted. Any net that is listed in the `SPECIALNETS` section of the DEF file is considered a special net. Mixed nets (nets that are listed partially in the DEF `SPECIALNETS` section and partially in the DEF `NETS` section) are also treated as special nets.

Note: The special nets marked as `USE POWER` or `USE GROUND` are not extracted even if you do not specify this parameter.

Default: All special nets, except those marked as `USE POWER` or `USE GROUND`, are extracted.

`-outputFileMaxSize` *`value`*

Specifies the desired maximum size of output files, in bytes. This value is used to split large Detailed Standard Parasitic Format (DSPF) or SPEF files into several smaller files when necessary because of the file size-handling constraints of downstream tools.

Type: Real

`-outputFileName` *`fileName`*

Specifies the name of the output file.

Default: *`opCellName`*`.spef`

`-outputIncompleteNets`

Includes open nets in the output DSPF or SPEF file.

`-outputUnconnectedPins`

Produces file containing a list of pins that are logically connected but not physically connected in the design.

`-pinChar` *`value`* Specifies the pin notation.

Type: String

`-promotePinPad {logical | physical | none}`

Determines what parasitic data to extract for a top-level logical pin connected to a pad pin by a net with no routing.

Default: `none`

Choose one of the following:

`logical`	Constructs the logical resistance and capacitance network for the connection between the top-level logical pin and the pad pin but sets the parasitic data to zero.
`none`	Ignores the top-level pin and extracts no parasitic data.

`-rcType {coupledRc | decoupledRc}`

Specifies the resistance and capacitance network to extract.

Default: `coupledRC`

Choose one of the following:

`coupledRc`	Extracts the coupled resistance and capacitance network.
`decoupledRc`	Extracts the decoupled resistance and capacitance network.

`-relativeCThreshold` *`value`*

Sets a percentage threshold value that determines when the extractor lumps a net's coupling capacitance to ground and when it records the coupling capacitance for the net.
Default: `0.01` (1 percent)
Type: Real
Range: `0-0.2`

`-spefOutput [extended | generic | PrimeTime | starN | CeltIC | none]`

Writes the qx output file specified by the variable *`outputFileName`*, in reduced or unreduced SPEF format.

Default: `none`

Choose from one of the following:

`extended`	Includes resistor width and length information in the SPEF.
`generic`	Includes the RC network in the output file.
`PrimeTime`	Includes only the RC network between cells in the output file for input to Synopsys PrimeTime timing verifier.
`starN`	Prints the internal nodes (*N) in the connectivity section of the SPEF output.
`CeltIC`	Includes the RC network between cells in the output file for input to Cadence CeltIC NDC signal integrity analysis tool.
`none`	Does not generate a SPEF file.

The `generic`, `CeltIC`, and `starN` options produce the same results, and generate the driving cell (*D) for all type of pins: input, output, and bidirectional.

`-techFile` *`tech_file`*

Specifies the name of the QRC technology file.

`-totalCThreshold` *`value`*

Sets a fixed threshold value that determines when the extractor lumps a net's coupling capacitance to ground and when it records the coupling capacitance of the net. If the total capacitance of a net is larger than this value, the extractor might keep the net's coupling capacitance depending on the other threshold parameter settings. If the total capacitance is smaller than this value, the extractor lumps the coupling capacitance to ground. The extractor uses this parameter only during coupled extraction.

Type: Real

Examples

- The following command extracts RC parasitics using technology file `qrc.tch`, and cell library file `library_new`, and generates a reduced SPEF file `qrc.spef`:

```
runQRC -outputFileName qrc.spef -techFile qrc.tch
   -libraryName ./library_new
```

- The following command extracts RC parasitics using command file `qrc.cmd`:

```
runQRC -cmd qrc.cmd
```

Sample Command File

The following is a typical format of a command file:

```
extraction_type all coupled_rc
setvar library_name library
setvar tech_file qrc.tch
setvar output_file_name TOPCHIP_SP
setvar gray_data gds
setvar spef_output generic
```

setExtractRCMode

```
setExtractRCMode
     [-help]
     [-reset]
     [-assumeMetFill scalevalue]
     [-capFilterMode { relOnly | relAndCoup | relOrCoup }]
     [-compressOptMemRCDB {true | false}]
     [-coupled {true | false}]
     [-coupling_c_th value]
     [-defViaCap {true | false}]
     [-effortLevel { medium | high }]
     [-engine {default | detail | CCE}]
     [-extraCmdFile fileName]
     [-force {true | false}]
     [-hardBlockObs {true | false}]
     [-incremental {true | false}]
     [-ipdb dirName]
     [-noGroundXCap {true | false}]
     [-noReduce {true | false}]
     [-optMemory {true | false | auto}
     [-rcdb {fileName}]
     [-reduce value]
     [-relative_c_th value]
     [-scOpTemp value]
     [-specialNet {true | false}]
     [-total_c_th value]
     [-useLEFCap {true | false}]
     [-useLEFResistance {true | false}]
     [-useNDRForClockNets {true | false}]
     [-useShieldingInDetailMode {true | false}]
     [-viaCap {true | false}]
```

Sets the native RC extraction mode. Use this command before using the `extractRC` command. You can use the `setExtractRCMode` command multiple times during an Encounter session to specify additional parameters or change the previously defined values for default, detail, or CCE mode. You can perform RC extraction in default, detail, or CCE mode. Each mode supports a specific set of `setExtractRCMode` command parameters. The following table lists the extraction mode support for each parameter.

Important

In previous releases, the `setExtractRCMode` command accepted incomplete parameter names. Now, only complete parameter names are recognized by the software. If you enter an incomplete parameter name, the software will return an error that the given parameter is not a legal argument. To avoid this error, update your scripts to use the legal (documented) parameter syntax.

setExtractRCMode Parameters	RC Extraction Mode		
	default	**detail**	**CCE**
-assumeMetFill	X	X	
-capFilterMode		X	X
-compressOptMemRCDB (-optMemory true only)		X	X
-coupling_c_th (-coupled true only)		X	X
-coupled		X	X
-defViaCap	X		
-effortLevel			X
-extraCmdFile			X
-force		X	
-hardBlockObs		X	
-ipdb		X	
-noGroundXCap		X	
-noReduce		X	
-optMemory		X	X
-rcdb		X	
-reduce		X	
-relative_c_th		X	X
-reset	X	X	X
-scOpTemp	X	X	X
-specialNet	X	X	
-total_c_th		X	X

setExtractRCMode Parameters	RC Extraction Mode		
	default	detail	CCE
-useLEFCap	X	X	
-useLEFResistance	X	X	
-useNDRForClockNets	X		
-useShieldingInDetailMode		X	
-viaCap		X	

Parameters

`-assumeMetFill` *`scalevalue`*

Takes into account the metal fill in the design.

In detailed extraction, *`scalevalue`* is not required and will be ignored. The detailed extraction uses the `FILLACTIVESPACING` values from LEF to account for metal fill in the design. If the `FILLACTIVESPACING` values are not specified in the LEF file, the software uses the default values.

In default extraction, when calculating the capacitance of a wire segment in layer *`N`*, this parameter always assumes layer *`N-1`* and layer *`N+1`* to be filled. Specify one of the following values:

- `0`: Assume no metal fill in layer *`N`*.
  - When you do not specify this parameter, the software assumes no metal fill in layer *`N`*. This is equivalent to setting the *`scalevalue`* parameter to `0`.
- `1`: Assume full metal fill at minimum spacing distance.
  - Where there is a minimum of signal-to-signal spacing, the cross-coupling capacitance will be extracted and reported.
  - Where there is no wire within the minimum spacing, a capacitance to ground will mimic minimum-spaced metal fill in that location.
- *`x`*: Capacitance value is calculated once with full metal fill and once without metal fill. The user-specified scale value (decimal value between `0 and 1`) determines the interpolation point between the two calculated values, for example, `0.5`

Default: 0

`-capFilterMode { relOnly | relAndCoup | relOrCoup }`

Specifies the capacitance filtering mode. The filtering modes are conditions which define how to use the `-coupling_c_th` and `-relative_c_th` values, or their combination. Based on the filtering mode specified, the software determines the nets in the design, which will have their coupling capacitance lumped to the ground.

Note: The behavior of the `-total_c_th` filter is independent of the `-capFilterMode` parameter setting. Nets that have a total capacitance value lower than the value specified with the `-total_c_th` parameter will have their coupling capacitances grounded regardless of the `-capFilterMode` setting.

You can use one of the following modes:

- `relOnly`: Grounds the coupling capacitance of nets based on the threshold value set by the `-relative_c_th` parameter.
- `relAndCoup`: Grounds the coupling capacitance of nets only when it is lower than the threshold value specified with the `-relative_c_th` parameter as well as the `-coupling_c_th` parameter. In this mode, you enforce maximum restriction on grounding the coupling capacitances.
- `relOrCoup`: Grounds the coupling capacitance of nets if it is lower than the threshold value specified with one of the `-relative_c_th` or `-coupling_c_th` parameters.

Default: For CCE extraction, the default value is `relAndCoup`. For detailed extraction, the default value is dependent on the process node specified with the `setDesignMode -process` command. If the process node is 130nm or below, the default value is `relAndCoup` and if the process node is above 130nm, the default value is `relOnly`.

Important: You can specify this parameter when using the detailed or CCE extraction modes only.

`-compressOptMemRCDB {true | false }`

Generates compressed RCDB data while using the direct access mode (`-optMemory true`).

Default: `false`

Note: This functionality reduces the disk usage but leads to marginal increase in run time.

`-coupled {true | false}`

Outputs both total and cross-coupled capacitance. The `-coupled` false value is typically used for static timing analysis.

Default: `true` (for detailed and CCE extraction)

`-coupling_c_th` *`value`*

Specifies the threshold value that determines when the extractor lumps a net's coupling capacitance to ground. This parameter is applicable only when the `-coupled` parameter is set to `true`.

The software decouples the coupling capacitance of nets when the total coupling capacitance between the pair of nets is lower than the threshold specified with this parameter. The actual decoupling also depends on the setting of the `-capFilterMode` parameter.

Default: `3 fF`. If the `setDesignMode -process` *`process_node`* command is used, the software assigns the recommended default value for this parameter based on the process node specified.

 *Tip*

See the `setDesignMode` command for the list of default values that are applied based on the process node setting.

`-defViaCap {true | false}`

Takes into account the via capacitance in the default RC extraction results. Specify a value of `true`, if you are experiencing correlation issues caused by deviations in clock latency when running default RC extraction.

Default: `false`

`-effortLevel { medium | high }`

Specifies the CCE variant that you want to use for performing RC extraction.

Default: `high`

You can use this parameter only with the `setExtractRCMode -engine CCE` setting.

This parameter supports the following values:

`medium` Invokes the fastCCE extraction mode. fastCCE is a faster but slightly less accurate variant of CCE extraction. However, the fastCCE extraction mode is more accurate compared to the detailed extraction mode.

Note: This setting does not require a QRC license.

`high` Invokes the CCE extraction engine. This is the same as specifying the `-engine CCE` setting.

Important Considerations

- The `optDesign -si` command honors the `-effortLevel` parameter.
- The `optDesign -postRoute` command currently does not recognize CCE as an engine choice regardless of whether you use the `-effortLevel` parameter.

`-engine {default | detail | CCE }`

Specifies the extraction mode to use.

Default: `default`

`default` Uses default mode.

RC extraction is done by the fast density measurements of the surrounding wires; coupling is not reported.

Use this mode to perform extraction on pre-routed designs.

`detail` — Uses detailed mode.

RC extraction is done by the detailed measurement of the distance to the surrounding wires; coupling is reported.

`CCE` — Uses the Cadence Common Extraction (CCE) engine in full-chip or incremental extraction modes. Encounter determines which mode to use based on the extent of the logical and physical changes in the design.

CCE provides better quality extraction compared to the detailed mode and gets best correlation with the standalone QRC sign-off mode.

In addition, CCE supports distributed processing. For example, you can specify the following commands to perform distributed processing:

```
setDistributeHost -rsh -add \
{ host1 host 2 host 3 }
setMultiCpuUsage -numHosts 3
setExtractRCMode -engine CCE
extractRC
```

CCE does not support multi-threading and super-threading operations. Therefore, the following options of the `setMultiCpuUsage` command are not supported:

- `-numThreads`
- `-superThreadsNumThreads`
- `-superThreadsNumHosts`

`-extraCmdFile` *`fileName`*

Specifies the QRC extra command file for CCE extraction. The extra command file supports the following variables:

- `setvar dump_options {`**`true`**` | false}`
- `setvar copy_port_to_obs {`**`true`**` | false}`
- `setvar metal_fill_type {grounded |` **`floating`** `| virtual}`
- `setvar compress_cache_file {true |` **`false`**`}`

Note: The **bold** text indicates that these values are used by default when the commands are not explicitly specified in the extra command file.

`-force {true | false}`

Runs extraction without checking if the placement data or RC parameters have changed.

Default: `false` (Checks to see if the placement data and other RC parameters, such as scale factor have changed in the current setting. If there is no change, the command does not perform extraction.)

`-hardBlockObs {true | false}`

Makes cell geometries on metal layers from obstruction, and power and ground pin shapes visible to the extractor for hard blocks only. The software recognizes the following three types of hard blocks:

- Hard macro (`CLASS BLOCK` in LEF file)
- Blackbox (`CLASS BLACKBOX` in LEF file)
- Partition (Committed during Encounter session)

Specify a value of `true` when routing goes over the listed hard blocks. A value of `true` removes the optimism for such routes, and provides better correlation with sign-off extraction without significantly impacting runtime.

Default: `false`

`-help`

Outputs a brief description that includes type and default information for each `setExtractRCMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command:
`man setExtractRCMode`.

`-incremental {true | false}`

Uses incremental CCE extraction mode.

If incremental mode is set, after initial extraction on full design is performed, CCE extraction checks for design changes and runs extraction in incremental mode on the modified areas only. This behavior, however, is applied by the software based on the extent of design changes on the premise that there will be run time improvement.

Incremental mode provides significant performance improvement during signal integrity optimization phase.

Default: `true`

`-ipdb` *`dirName`*

Uses the interconnect parasitic database (ipdb) which allows faster direct access instead of the rcdb. The *`dirName`* variable is the name of the directory. It must not have an `.ipdb` extension.

`-noGroundXCap {true | false}`

Does not ground smaller coupling capacitances, to save on run time. If you specify a value of `false`, smaller coupling capacitances are grounded in a post-processing step after extraction, to honor the specified coupling capacitance threshold settings.

You can specify a value of `true` in a script to temporarily overwrite the threshold handling without changing the actual capacitance threshold settings.

Default: `false`

Important

Option `-noGroundXCap` is obsolete. By default, in coupled mode (`-coupled true`), the software grounds the smaller coupling capacitances based on the coupling capacitance threshold settings specified using the `-total_c_th`, `-relative_c_th`, and `-coupling_c_th` options. In detailed extraction mode, note that the `-coupling_c_th` option is considered only when the `-capFilterMode` option is set to `relAndCoup` or `relOrCoup`. If you do not want to ground coupling capacitances in coupled mode, set the coupling capacitance threshold values to zero.

`-noReduce {true | false}`

Does not perform RC reduction on the output file.

Default: `false`

Important

Option `-noReduce` is obsolete and is no longer recommended. The software does not perform RC reduction by default.

`-optMemory {true | false | auto}`

Specifies the software to optimize memory usage during RC extraction.

When set to `false`, the software has faster runtime but the memory is not optimized. This parameter might be useful in smaller designs.

When set to `true`, the memory is optimized. This ensures that the software doesn't run out of memory for large designs. This parameter, however can result in increase in runtime.

When set to `auto`, the software automatically switches to using the memory optimized mode from faster runtime mode during detailed or CCE extraction if the design size is more than 600K nets on 32bit machine or more than 1500K on 64bit machine.

Default: The `auto` parameter is used.

Note: The software always uses memory optimized mode, equivalent to `-optMemory` parameter set to `true` during multi-mode multi-corner analysis.

`-rcdb {`*`fileName`*`}` Specifies the RC database output filename.

Important

Parameter `-rcdb` is obsolete and is no longer recommended. The software generates the RC database (RCDB) in the current working directory by default unless you have specified an alternative location using the `FE_TMPDIR` or `TMPDIR` variable.

`-reduce` *`value`* Specifies the threshold value (in picoseconds) used to perform RC reduction.

Default: `5`

Important

Option `-reduce` is obsolete and is no longer recommended. The software does not perform RC reduction by default.

`-relative_c_th` *`value`*

Sets a ratio threshold value that determines when the extractor lumps a net's coupling capacitance to ground. This parameter is applicable only when the `-coupled` parameter is set to `true`.

If the total coupling capacitance between a pair of nets is less than the percentage (specified with this parameter) of the total capacitance of the net with the smaller total capacitance in the pair, the coupling capacitance between these two nets will be considered for grounding. The actual decoupling also depends on the setting of the `-capFilterMode` parameter.

Default: `0.03`. If the `setDesignMode -process` *`process_node`* command is used, the software assigns the recommended default value for this parameter based on the process node specified.

Tip

See the `setDesignMode` command for the list of default values that are applied based on the process node settings.

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setExtractRCMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-scOpTemp` *`value`*

Specifies the temperature derating value to be used in single-corner analysis mode. The derating value is specified in centigrade.

Default: `25C`

`-specialNet {true | false}`

Extracts special nets, including nets that are a combination of signals and special wires. You can specify this option in default and detail mode of extraction. Special nets are always extracted in CCE extraction mode.

Default: `false`

`-total_c_th` *`value`*

Specifies the threshold value (femtoFarads) that determines when the extractor lumps a net's coupling capacitance to ground. This parameter is applicable only when the `-coupled` parameter is set to `true`.

The software grounds the coupling capacitances for the nets which have a total capacitance value less than the value specified with this parameter.

Default: `5 fF`. If the `setDesignMode -process` *`process_node`* command is used, the software assigns the recommended default value for this parameter based on the process node specified.

Tip

See the `setDesignMode` command for the list of default values that are applied based on the process node settings.

`-useLEFCap {true | false}`

Forces extraction to use the `CPERSQDIST` and `EDGECAPACITANCE` values defined in LEF, instead of the values in the capacitance table. If the `CPERSQDIST` values are missing for layers, the software ignores this parameter and uses the capacitance table instead.

This parameter is available in both detail and default mode. With this parameter, the software ignores the effects of neighboring nets. Therefore use this parameter to achieve better correlation with the parasitic estimators that only use layer assignment, length, and width to calculate the capacitance of the wires.

Default: `false`

`-useLEFResistance {true | false}`

Forces extraction to use the resistivity coefficients for the layers, and via resistances defined in the LEF file, instead of the values defined in the capacitance table. When you specify `-useLEFResistance true`, the software uses the drawing width for calculating the resistance. The calculation is independent of any bias or edge specified in the capacitance table.

Default: `false` (The value in the capacitance table takes precedence over the value in the LEF file.)

`-useNDRForClockNets {true | false}`

Uses the non-default rules from LEF to calculate approximate shielding and spacing values, instead of using the density values.

Default: `true`

`-useShieldingInDetailMode {true | false}`

Assumes shielding for any wire segment of a clock net that does not have any shielding yet, and the space between the wire segment and its neighbor is greater than or equal to the following:

`2s+w`

where,

s = minimum spacing or non-default spacing

w = minimum width or non-default width

This parameter is available in detailed mode only.

Default: `false` (RC extraction supports the non-default rules from the LEF file for all clock nets using `-useNDRForClockNets true`. If you do not want to use the non-default rules for clock nets, you must set `-useNDRForClockNets false`, which uses the general design density values to calculate the approximate shielding and spacing values.)

`-viaCap {true | false}`

Extracts capacitance values considering part of the via geometries that do not overlap with the wire geometries. Via geometries that do not overlap the wire geometries are often found if you are designing using newer design technologies such as 90nm or when using multiple cut via.

Default: `true`

Examples

- The following command includes special nets in extraction:

```
setExtractRCMode -engine detail -specialNet true
```

- The following command uses the capacitance table in RC extraction (preceded by a `readCapTable` command):

```
setExtractRCMode -engine detail
extractRC
rcOut -spef fe_extended.spef
```

- The following is a typical command sequence for the metal fill parameter:

```
setExtractRCMode -engine detail -assumeMetFill 1
```

```
extractRC
```

- The following command uses the resistivity coefficients for the layers and via resistances defined in the LEF file:

```
readCapTable test.captable
setExtractRCMode -engine detail -useLEFresistance true
extractRC
```

- The following command resets the `-relative_c_th` parameter to its default value:

```
setExtractRCMode -reset -relative_c_th
```

- The following command resets all `setExtractRCMode` parameters to their default values:

```
setExtractRCMode -reset
```

Related Topics

- SI Closure in the *Encounter Flat Implementation Flow Guide*
- Signoff in the *Encounter Flat Implementation Flow Guide*

setQRCTechfile

```
setQRCTechfile
     techfileName
     [-opTemp temp]
     [-layermap layermapfile]
```

Specifies the QRC technology file, operating temperature, and additional layer map file for CCE extraction. You can obtain the values for these parameters using the getQRCTechfile command.

Note: Use this command for single corner extraction only. For multi-corner extraction, use the following command:

```
create_rc_corner -name rcCornerName -qx_tech_file fileName
```

Parameters

-layermap *layermapfile*

Specifies the location of the optional layer map file. Layer mapping between the LEF or DEF file and the QRC technology file is normally done automatically. This parameter is only necessary if this mechanism fails. If necessary, you can use a copy of the automatically generated layer map file `extLogDir/`*`design`*`.layermap.log` and modify it for your own use.

Note: If used, the layer map file must be in the QRC LibGen format.

The following is an example of the layer map file format:

```
#<Layer Type>  <QRC Layer>  <Fixed Keyword>  <LEF Layer>
  poly            polycide        lefdef        Poly
  via              cont           lefdef        Cont
  metal            metal_1        lefdef       Metal1
  via              via_1          lefdef        Via12
  metal            metal_2        lefdef        Metal2
```

-opTemp *temp*

Specifies the operating temperature. The software derates the value of resistance based on this temperature value. The unit of temperature is Celsius.

techfileName Specifies the name of the QRC technology file. The technology file contains the interconnect models that CCE extraction uses to extract the interconnect parasitic capacitance and resistance.

Note: CCE extraction does not support technology files created with the icecaps program. However, it supports cell-level only and Unified QRC Techfile.

Example

- The following commands perform CCE extraction using the `tech090_5lm.tech` technology file:

```
setQRCTechfile tech090_5lm.tech
setExtractRCMode -CCE
extractRC
```

setRCFactor

```
setRCFactor
     [-defcap scaleFactor]
     [-defres scaleFactor]
     [-detcap scaleFactor]
     [-detres scaleFactor]
     [-xcap scaleFactor]
     [-res scaleFactor]
```

Sets scale factors for the resistors and capacitors that are extracted in either default or detail extraction mode. Use this command before running the `extractRC` command and after running the `generateCapTbl` command.

The Encounter software uses these scale factors to multiply the extracted resistor values, the extracted total capacitor values, and the cross-coupling capacitor values. The default value for all scale factors is `1.0`.

An appropriate scaling factor can be set to increase the correlation between Encounter and sign-off extraction. The scaling factors can be set by comparing extraction parasitics files using utility tools like Ostrich, which is available in the Encounter hierarchy. The best scaling factor might depend on the technology and the density of the design. You can also use the `getRCFactor` command to automatically find and set the optimal scaling factor.

Note: When using a comparison script to set the resistor scaling factor make sure both extractors did not perform any RC reduction. Check the via resistance and sheet resistance used by both tools. When using a comparison script to set the coupling scaling factor make sure no threshold value was used for grounding the smaller coupling capacitance.

To report the RC scale factors, use this command without specifying any of the parameters.

Parameters

`-defcap` *`scaleFactor`*

Specifies the capacitance scale factor for RC Extraction in default mode.
Default: `1.0`

`-detcap` *`scaleFactor`*

Specifies the capacitance scale factor for RC extraction in detail mode.
Default: `1.0`

`-defres` *`scaleFactor`*

Specifies the resistance scale factor for RC extraction in default mode.

Default: When this parameter is not specified, the value specified with the `-res` parameter is used for default extraction.

Note: When used, this parameter overrides the value specified with the `-res` parameter.

`-detres` *`scaleFactor`*

Specifies the resistance scale factor for RC extraction in detail mode.

Default: When this parameter is not specified, the value specified with the `-res` parameter is used for detailed extraction.

Note: When used, this parameter overrides the value specified with the `-res` parameter.

`-res` *`scaleFactor`* Specifies the resistance scale factor to be applied in default and detailed extraction modes.
Default: `1.0`

Important

Parameter `-res` is obsolete and has been replaced by the `-detres` and `-defres` parameters.

`-xcap` *`scaleFactor`* Specifies the cross-coupling capacitance scale factor.
Default: `1.0`

Example

- The following command sets the capacitance scale factor to `0.81` in RC extraction default mode and the resistance scale factor to `1.1`:

```
setRCFactor -defcap 0.81 -res 1.1
```

Related Topics

- timing.tcl in the *Encounter Flat Implementation Flow Guide*

setShrinkFactor

`setShrinkFactor` *`shrinkFactor`*

Sets the value by which the Encounter software shrinks the design before extraction.

The shrink factor is used only for extraction calculations. It does *not* change the actual design.

Note: You do not need to specify the shrink factor if the capacitance table file in use contains the shrink factor specififcation in its header.

Parameter

`shrink_factor`	Specifies the value by which the Encounter software shrinks a design. *Default:* `1.0`

Example

The following command sets the shrink factor to 0.9:

```
setShrinkFactor 0.9
```

Setting the shrink factor to 0.9 reduces the wire lengths, widths, and spacings in the design by 0.9 times the original values.

spefIn

```
spefIn
     [-dumpMissedNet mnFileName]
     list_of_fileNames
     [-rc_corner rcCornerName spefFileName]
     [-starN | -noStarN]
     [-XCapFilter spefFileName]
     [–sspef file_name]
```

Loads resistors and capacitors for the interconnects in SPEF into the Encounter database to calculate delays or build a timing graph. No placement or routing data is needed. Use this command to read in a SPEF file generated by a third-party extraction tool.

Note: You can use this command to load a SPEF file, which is compressed (.gzip) format.

Parameters

`-dumpMissedNet` *`mnFileName`*

Writes a detailed listing of all the missing nets to the file *`mnFileName`*.

To support multi-corner analysis, you can use this parameter in conjunction with the `-rc_corner` parameter to generate a separate missing nets file for each active RC corner. The software uses the RC corner ID as the prefix for each RC corner file name.

Note: This parameter is now obsolete. The software by default prints the missing nets in file *`TopCellName`*`.missing_nets.rpt` file.

`list_of_fileNames`

Specifies a SPEF file or list of SPEF files as input for flat chip or hierarchical SPEF stitching respectively.

Note: If you are loading more than one SPEF file, you must specify the top-level SPEF file as the last entry in the list.

`-rc_corner` *`rcCornerName spefFileName`*

Annotates the parasitics to the predefined RC corner for multi-corner analysis. If you want to use SPEF information when calculating delays in multi-mode multi-corner analysis mode, you must annotate the parasitics for each RC corner in the design.

For example, if you have four active RC corners in the design, you *must* annotate the parasitics for each RC corner.

```
spefIn -rc_corner corner1 rc1.spef
spefIn -rc_corner corner2 rc2.spef
spefIn -rc_corner corner3 rc3.spef
spefIn -rc_corner corner4 rc4.spef
```

`-sspef` *`file_name`* Reads the statistical SPEF file. SSPEF file is required to get interconnect sensitivities that are used to consider interconnect variations in statistical static timing analysis (SSTA).

`-starN | -noStarN`

Specifies whether or not to read the `*N` statements in the SPEF file. The `*N` statements define the location of the RC nodes. The `-starN` parameter reads the `*N` statements and the `-noStarN` parameter ignores the `*N` statements.

`-XCapFilter` *`spefFileName`*

Removes (filters) tiny cross-coupling capacitances from the specified SPEF file before loading it into the internal RC database. By default, no filtering is performed.

Note: This parameter is provided for backward compatibility with Encounter versions where filtering was default.

Examples

- The following command loads the SPEF file `TOPCHIP_SP.spef` for a flat chip methodology:

  ```
  spefIn TOPCHIP_SP.spef
  ```

- The following command loads multiple SPEF files for hierarchical SPEF stitching:

  ```
  spefIn { PTN/results_conv/results_conv.spef tdsp_core/tdsp_core/
  tdsp_core.spef PTN/TOPCHIP_SP/TOPCHIP_SP.spef }
  ```

wireload

```
wireload
     [-outfile fileNamePrefix]
     [-scale factor]
     [-percent samplingRate]
     [-cell cellName | -inst instanceName [-name modelName]]
     [-instanceBased]
     [-cellLimit moduleSize]
     [-custom]
     [-noSmooth]
     [-netCoverRatio]
     [-logarithmic]
     [-peak]
     [-view viewName]
```

Generates the wireload models and outputs the models in hierarchical and flat formats. Use this command after extracting the RC data in your design.

The wireload models in the hierarchical format are generated for cells and instances based on external nets (hierarchical view). The hierarchical format considers only nets that crosses the boundaries of child modules within the module, while the flat format considers all nets within the module and its child modules.The wireload models in the flat format are generated for cells and instances based on internal nets (flat view).

Two wireload model formats are available: standard or custom.

Parameters

`-cell` *`cellName`*
: Specifies the cell name for generating a wireload model.

`-cellLimit` *`moduleSize`*
: Specifies the module size limit of cells or instances for generating a wireload model. Cells or instances that contain fewer than the *`moduleSize`* cells or instances will not have wireload models generated.
Default: `100`

`-custom`
: Generates the Synopsys custom wireload model format, which is also known as the long format. The default is the Synopsys standard wireload model format, which is also known as the short format.

`-inst` *`instanceName`*	Specifies the name of the hierarchical instance in a path for which to generate a wireload model. The hierarchical instance can be at any point in the path or at the beginning or end of the path.
`-instanceBased`	Generates wireload models for each instance in the design. The default is to generate wireload models for each cell type used in the design.
`-logarithmic`	Uses logarithmic average of the fanout value to calculate the wireload model.
`-name` *`modelName`*	Renames the cell name or instance name.
`-netCoverRatio`	Specifies the percentage of nets to be considered for generating wireload model.
`-noSmooth`	Suppresses the smoothing of the wireload models.
`-outfile` *`fileNamePrefix`*	Specifies the filenames for the generated wireload models. The Encounter software generates four files for the hierarchical and flat models, and one command file each to load the model files back into the synthesis tool.
`-peak`	Uses peak fanout value to calculate the wireload model.
`-percent` *`samplingRate`*	Specifies the sample rate of the number of nets for calculation of wireload models. The values are between `0.0` and `1.0` (all sampled). *Default:* `1.0`
`-scale` *`factor`*	Specifies the scaling value used to multiply the results. *Default:* `1.0`.
`-view` *`viewName`*	Generates the wireload models for the specified analysis view. *Default*: Uses the default analysis view.

Example

- The following command generates instance-based wireload models and outputs two model files, `design_name.wl.flat` and `design_name.wl.hier` and two command files for loading:

  ```
  wireload -instanceBased
  ```

28

RTL Synthesis Commands

The Encounter RTL synthesis commands provide access to the features of Encounter™ RTL Compiler Ultra (RC) which are used to generate a gate-level netlist from an RTL Netlist.

The following information describes the commands and options that control the generation of the gate-level netlist.

compileDesign

```
compileDesign
     [-script fileName]
     [-outDir dirName]
     [-removeTempFiles]
     [-setupOnly]
     [-noCommit]
     [-noCostGroup]
```

Sets up, launches, and controls the synthesis run. The `compileDesign` command uses parameters specified by setCompileMode on page 1218.

Parameters

`-outDir` *`dir_name`*	Specifies the output directory for RC results.
`-setupOnly`	Creates directories and files as needed to prepare the RC synthesis run, without actually running the synthesis step.
`-removeTempFiles`	Removes all intermediate files, including the RC script, after synthesis is done.
`-noCommit`	Runs RC synthesis and generates the gate-level netlist, but does not reload the synthesized netlist. This allows you to iterate synthesis runs to refine the netlist results.
`-noCostGroup`	Disables creating cost group in the RC synthesis run, based on the I/O's and registers. Specify this parameter if your design does not have any registers.
`-script` *`file_name`*	Specifies the name of the RC script containing the commands used to run and control synthesis.

Example

```
loadRTLConfig extendedRTL.conf
setCompileMode [-synEffort high]
getCompileMode
compileDesign
```

getCompileMode

```
getCompileMode [-mode_name...]
```

Returns the current value(s) for the modes specified by *mode_name* or all modes when no mode specified. This command displays the information specified by the setCompileMode command.

Parameters

-mode_name ...	Returns the values specified by the setCompileMode command.

loadRTLConfig

`loadRTLConfig` *`extended_config_file`*

Loads in the extended configuration file and sets the necessary internal global variables. This command does not start any processes.

Parameters

`extended_config_file`

Specifies the name of the extended configuration file.

saveRTLConfig

`saveRTLConfig` *`extended_config_file`*

Saves the RTL values into an extended configuration file.

Parameters

`extended_config_file`

Specifies the name of the extended configuration file.

setCompileMode

```
setCompileMode
     [-synEffort low|medium|high]
     [-mapEffort low|medium|high]
     [-incrEffort low|medium|high]
     [-opCond opcond_name]
     [-wlmName wireload_name]
     [-wlmLibrary library_name]
     [-wlmMode wireload_mode]
     [-rcVar {name_value_pair} ...]
     [-rcAttr {name_value_pair} ...]
     [-preloadInclude file_name ...]
     [-postloadInclude file_name ...]
     [-endInclude file_name ...]
     [-report report_type...]
```

Controls all the active commands for synthesis in RC. The parameters that you specify with this command are used by compileDesign on page 1214.

Parameters

`-endInclude file_name ...`

Specifies the name of a file to be included after synthesis and optimization. For example, you can create a file to report on timing paths after synthesis to compare with pre-synthesis data.

`-incrEffort [low|medium|high]`

Specifies the level of effort for incremental synthesis. Incrementally optimizes mapped gates. Allows the mapper to preserve the current implementation of the design and perform incremental optimizations only if the procedure guarantees an improvement in the overall cost of the design. This parameter only works with an already mapped design.

Low effort performs very little optimization, incremental clean up, or redundancy identification and removal. The low setting is generally not recommended.

Medium effort performs better timing-driven structuring, incremental synthesis, and redundancy identification and removal on the design. Medium is the default.

High effort performs the timing-driven structuring on larger sections of logic and spends more time and makes more attempts on incremental clean up. This effort level involves very aggressive redundancy identification and removal.

`-mapEffort [low|medium|high]`

Specifies the level of effort for mapping and logic optimization. The aim of the optimization is to provide the smallest possible implementation of the synthesized design that still meets the supplied timing goal.

Low effort performs very little optimization, incremental clean up, or redundancy identification and removal. The low setting is generally not recommended.

Medium effort performs better timing-driven structuring, incremental synthesis, and redundancy identification and removal on the design. Medium is the default.

High effort performs the timing-driven structuring on larger sections of logic and spends more time and makes more attempts on incremental clean up. This effort level involves very aggressive redundancy identification and removal.

`-opCond` *`opcond_name`*

Overwrites the default operating condition. If no default is defined, this parameter needs to be specified.

`-preloadInclude` *`file_name ...`*

Specifies the name of a file to be included prior to loading the design. For example, you can create files for variable and attribute definitions.

`-postloadInclude` *`file_name ...`*

Specifies the name of a file to be included after loading the libraries, RTL, and constraints. For example, you can create a file with commands to report specific timing paths before synthesis.

`-rcAttr {`*`name value object`*`} ...`

Specifies the names, values, and objects for a list of RC attributes.

`-rcVar {`*`name value`*`}` ...

Specifies the names and values for a list of RC attributes.

`-report` *`report_type`* ...

Specifies the type of the reports to be written during the synthesis run. The report options are `all`, `area, design, exception, gates, none, timemem, timing, timingdetail, zero`.

`-synEffort [low|medium|high]`

Specifies the level of effort for generic RTL optimization.

Low effort performs very little optimization, incremental clean up, or redundancy identification and removal. The low setting is generally not recommended.

Medium effort performs better timing-driven structuring, incremental synthesis, and redundancy identification and removal on the design. Medium is the default.

High effort performs the timing-driven structuring on larger sections of logic and spends more time and makes more attempts on incremental clean up. This effort level involves very aggressive redundancy identification and removal.

`-wlmLibrary` *`library_name`*

Specifies the library containing the wireload model.

`-wlmMode` *`wireload_mode`*

Specifies how the wireload model is applied:

top: Uses the wire-load model of the top-level design for all nets in all subdesigns. Hierarchical boundary pins are not counted as fanouts.

enclosed: Uses the wire-load model of the smallest block that fully encloses the net to compute the load of the net. Hierarchical boundary pins are not counted as fanouts.

segmented: Divides nets that cross hierarchical boundaries into segments with one segment for each level of hierarchy. Separate load values are computed for each segment (counting the hierarchical boundary pins as individual fanouts) and the load values are added together.

`-wlmName` *`wireload_name`*

Specifies the name of the wireload model to be used during synthesis.

29

Route Commands

addChannelDensityControl

`addChannelDensityControl` *`llx lly urx ury layerName ratio`*

Controls the routing density for a specified area on a layer by setting a maximum density percentage that Trial Route cannot exceed.

Use the `addChannelDensityControl` command after performing placement.

Parameters

`layerName`	Specifies the layer on which to specify the routing density control.
`llx`	Specifies the lower left x coordinate of the density area.
`lly`	Specifies the lower left y coordinate of the density area.
`ratio`	Specifies the maximum routing density allowed for the area, as a percentage.
	Value: `0` (zero percent) to `1` (100 percent)
`urx`	Specifies the upper right x coordinate of the density area.
`ury`	Specifies the upper right y coordinate of the density area.

checkRoute

`checkRoute`

Checks the routing connectivity and reports the unconnected terminals, the number of open connections, and the number of dislocated and offgrid terminals.

Use the `checkRoute` command after routing with any of the routing commands in the Encounter software, or after loading routing data in Design Exchange Format (DEF) or Top Design Format (TDF).

createShield

```
createShield
     [-selected]
     [-include_fixed]
```

Creates shield wires for the NanoRoute router, after routing with the following `setNanoRouteMode` parameter set to `true`:

```
-routeDeferredShield
```

For more information, see `setNanoRouteMode`.

After creating the shields, the router generates a report that includes shielding coverage statistics.

Parameters

`-selected`	Creates shield wires for the selected nets. If you do not specify this option, all the clock nets will be shielding.
`-include_fixed`	Includes fixed wires for shielding. The wire will not be changed but the electrical characteristics can be altered by adding shielding so that this option is required to include the fixed wire for shielding.

Related Topics

- "Using the NanoRoute Router" chapter in the *Encounter User Guide*
  - Interpreting the Shielding Report

deleteShield

```
deleteShield {-nets list_of_signal_nets | -selected}
```

Deletes physical shielding wires from the specified or selected signal nets and removes SHIELDNET attributes from the nets.

Use the deleteShield command after setting the SHIELDNET attribute on a net.

To set the SHIELDNET attribute, use the setAttribute command.

Parameters

`-nets` *list_of_signal_nets*
: Specifies a list of signal nets with one or more SHIELDNET attributes. You can use the asterisk (*) wildcard in the net names.

`-selected`
: Lets you select nets.

Examples

The following command deletes SHIELDNET attributes and physical shielding wires from net clk1:

```
deleteShield -nets clk1
```

The following command deletes SHIELDNET attributes and physical shielding wires from nets c*:

```
deleteShield c*
```

The following command deletes SHIELDNET attributes and physical shielding wires from selected nets:

```
deleteShield -selected
```

describeCongestion

`describeCongestion`

Aggregates the congestion information into horizontal congestion and vertical congestion super gcells, and outputs a congestion report that provides an overall congestion measurement for the design and summarizes the super gcell overflow information.

Horizontal congestion is totalled into horizontal congestion super cells, which are tall, narrow boxes that typically have a height of four gcells and a width approximately equal to the height of a vertical congestion super gcell. Vertical congestion is totalled into vertical congestion super gcells, which are short, wide boxes that typically have a height of one gcell and a width approximately equal to the height of an horizontal congestion super gcell. All super gcells must be an integer number of gcells.

The report produced by the `describeCongestion` command differs from the default report generated by the `trialRoute` command in that it represents overflow congestion at a less detailed level than the finer gcell level. In addition, the report produced by `describeCongestion` reports the amount of overflow, whereas the report produced by `trialRoute` reports the number of remaining tracks.

For more information on the default Trial Route report, see "Congestion Distribution Report," in the *Encounter User Guide*.

You can use the `describeCongestion` command after running Trial Route.

Examples

The following example shows a congestion report output after using the `describeCongestion` command:

```
Gcells have been grouped into super gcells for coarser sampling.
There are 5248 that measure horizontal congestion.
There are 5248 that measure vertical congestion.

Overflow  numH           numV
-------------------------------------
      1:  135 ( 0.82%)   389 ( 2.36%)
      2:    0 ( 0.00%)    95 ( 0.58%)
      3:   29 ( 0.18%)   162 ( 0.98%)
      4:    0 ( 0.00%)    60 ( 0.36%)
      5:    0 ( 0.00%)    36 ( 0.22%)
      8:   44 ( 0.27%)     0 ( 0.00%)
      9:   15 ( 0.09%)     0 ( 0.00%)
```

```
overall congestion index: H 355 (2.15%), V 1004 (6.09%)
```

The following table defines the columns in the report:

Column	Definition
`Overflow`	The track supply minus the track demand.
`numH`	The number and percentage of stacks where the horizontal track demand exceeds the horizontal track supply by the given `Overflow` amount.
`numV`	The number and percentage of queues where the vertical track demand exceeds the vertical track supply by the given `Overflow` amount.

detailRoute

```
detailRoute
     [-select]
     [x1 y1 x2 y2]
```

Uses the NanoRoute® router to perform detailed routing on the entire design, an area of the design specified by the bounding box, or on selected nets. If you do not specify parameters, the router routes the entire design. If you specify both the bounding box and selected nets, the router routes only selected nets within the bounding box. You can perform detailed routing on specified area only if you are rerouting the design. If you have not already run detailed routing on the entire design, you cannot run detailed routing on a specified area.

This command supports multiple-CPU processing. For information, see Multiple-CPU Processing Commands and Accelerating the Design Process by Using Multiple-CPU Processing.

Use this command after completing the following steps:

1. Placing the design.
2. Routing special nets.
3. Setting net attributes with `setAttribute` and NanoRoute run-time options with `setNanoRouteMode`.

   For information on `setAttribute`, see “setAttribute” on page 1271. For information on `setNanoRouteMode`, see “setNanoRouteMode” on page 1280.

   Note: This step is optional. Complete this step only if you do not want to use the current values for attributes or default values for options.
4. Running global routing with `globalRoute`.

Parameters

`-select`	Routes selected nets. To select a net, use the `selectNet` command or click the net in the design display window. For information, see “selectNet” on page 556.

Tip

You can also specify `setNanoRouteMode -routeSelectedNetOnly true` to route the selected nets. For information, see "setNanoRouteMode" on page 1280.

x1 y1 x2 y2

Specifies the x and y coordinates of the bounding box.

Example

The following command selects `net1` and runs detailed routing on it:

```
selectNet net1
detailRoute -select
```

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - "Detailed Routing"

dumpCongestArea

```
dumpCongestArea {fileName | -all fileName}
```

Writes the routing congestion information to an ASCll output file. The output file reports the coordinates of overflow cells, vertical tracks, and horizontal tracks.

You can use the `dumpCongestArea` command after running Trial Route.

Parameter

`-all` *`fileName`*	Writes the congestion information for all gcells in the design to the specified output file.
`fileName`	Writes the congestion information for only the most congested gcells in the design to the specified output file.

Example

The following command writes the routing congestion information for all gcells to the `congest.txt` file:

```
dumpCongestArea -all congest.txt
```

dumpNanoCongestArea

```
dumpNanoCongestArea {fileName | -all fileName}
```

Writes the NanoRoute routing congestion information to an ASCII output file. The output file reports the total number of overflowed tracks, and the coordinates of overflow cells, vertical tracks, and horizontal tracks. The following example shows the information for a gcell in the ASCII output file:

```
(1192900, 974400) (1209700, 991200) V: -1/9/24 H: 0/16/39
```

The gcell has one vertical (V) track overflow (-1) from a layer. However, if all tracks are counted, it has nine remaining vertical tracks among 24 total tracks.

For horizontal (H) tracks, there was no overflow from any layer (0), and it has 16 remaining tracks among 39.

You can use the dumpNanoCongestArea command after running the NanoRoute globalRoute command.

Parameters

-all *fileName*	Writes the congestion information for all gcells in the design to the specified output file.
fileName	Writes the congestion information for only the most congested gcells in the design to the specified output file.

dumpNetsInCongestedArea

`dumpNetsInCongestedArea` *`fileName`*

Outputs the names of nets in the congested area of the design into the specified file, based on Trial Route results.

You can use the `dumpNetsInCongestedArea` command after running Trial Route.

Parameters

`fileName`	Specifies the name of the file in which to output the net names.

getAttribute

```
getAttribute -net netName
```

Displays the current net attribute settings in the Encounter console and in the Encounter log file. At the end of the display, the NanoRoute router lists each attribute, enclosing it in braces ({}) so you can use it in a Tcl script with variables. See "setAttribute" on page 1271 for descriptions of NanoRoute attributes.

Parameters

`-net netName`	Specifies the net for which to display attributes.

Example

The following command displays the attributes for `net1`:

```
getAttribute -net net1
```

The Encounter software returns the following:

```
#NET net1 attributes:
#    weight = 2 (default)
#    non_default_rule = default (default)
#    shield_net = none (default)
#    pattern = steiner (default)
#    top_preferred_routing_layer = 15 (default)
#    bottom_preferred_routing_layer = 0 (default)
#    preferred_extra_space = 0 (default)
#    avoid_detour = false (default)
#    skip_routing = false (default)
#    skip_antenna_fix = false (default)
#    si_post_route_fix = false (default)
{weight 2} {non_default_rule default} {shield_net none} {pattern steiner}
{top_preferred_routing_layer 15} {bottom_preferred_routing_layer 0}
{preferred_extra_space 0} {avoid_detour false} {skip_routing false}
{skip_antenna_fix false} {si_post_route_fix false}
```

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*

getNanoRouteMode

```
getNanoRouteMode
     [-dbReportWireExtraction]
     [-dbReportWireExtractionEcoOnly]
     [-dbSkipAnalog]
     [-drouteAntennaEcoListFile]
     [-drouteAutoCreateShield]
     [-drouteAutoStop]
     [-drouteCheckMinstepOnTopLevelPin]
     [-drouteElapsedTimeLimit]
     [-drouteEndIteration]
     [-drouteFixAntenna]
     [-droutePostRouteMinimizeViaCount]
     [-drouteMinLengthForWireSpreading]
     [-drouteMinLengthForWireWidening]
     [-drouteMinSlackForWireOptimization]
     [-drouteNoTaperInLayers]
     [-drouteNoTaperOnOutputPin]
     [-droutePostRouteLithoRepair]
     [-droutePostRouteSpreadWire]
     [-droutePostRouteSwapVia]
     [-droutePostRouteWidenWire]
     [-droutePostRouteWidenWireRule]
     [-drouteSearchAndRepair]
     [-drouteStartIteration]
     [-drouteUseBiggerOverhangViaFirst]
     [-drouteUseMultiCutViaEffort]
     [-drouteOnGridOnly]
     [-envAdvancedIntegration]
     [-envNumberFailLimit]
     [-envNumberWarningLimit]
     [-routeAllowPowerGroundPin]
     [-routeAntennaCellName]
     [-routeAntennaPinLimit]
     [-routeAutoGgrid]
     [-routeBottomRoutingLayer]
     [-routeConcurrentMinimizeViaCountEffort]
     [-routeDeferredShield]
     [-routeDeleteAntennaReroute]
     [-routeDesignFixClockNets]
     [-routeEcoOnlyInLayers]
     [-routeExtraViaEnclosure]
     [-routeFixTopLayerAntenna]
     [-routeHonorPartition]
     [-routeHonorPowerDomain]
     [-routeIgnoreAntennaTopCellPin]
     [-routeInsertAntennaDiode]
     [-routeInsertAntennaInVerticalRow]
     [-routeInsertDiodeForClockNets]
     [-routeMergeSpecialWire]
```

```
[-routeMinShieldViaSpan]
[-routeReverseDirection]
[-routeSelectedNetOnly]
[-routeSiEffort]
[-routeStrictlyHonorNonDefaultRule]
[-routeStripeLayerRange]
[-routeTdrEffort]
[-routeTopRoutingLayer]
[-routeUseBlockageForAutoGgrid]
[-routeWithEco]
[-routeWithLithoDriven]
[-routeWithLithoDrivenMargin]
[-routeWithSiDriven]
[-routeWithSiPostRouteFix]
[-routeWithTimingDriven]
[-routeWithViaInPin]
[-routeWithViaOnlyForStandardCellPin]
[-timingEngine CTE]
[-quiet]
```

Displays the following information about a `setNanoRouteMode` parameter in the Encounter log file and in the Encounter console:

- Option name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setNanoRouteMode` mode parameters.

Parameters

`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `setNanoRouteMode` for descriptions of the parameters you can specify.
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setNanoRouteMode` parameters in Tcl list format.

Examples

- The following command displays the current setting of the `-drouteFixAntenna` parameter:

  ```
  getNanoRouteMode -drouteFixAntenna
  ```

  The software displays the following information:

  ```
  -drouteFixAntenna true                  # bool, default=true
  true
  ```

- The following command displays the current setting for the `-drouteFixAntenna` parameter in Tcl list format only:

  ```
  getNanoRouteMode -quiet -drouteFixAntenna
  ```

  The software displays the following information:

  ```
  true
  ```

- The following command displays the current settings for all `setNanoRouteMode` parameters:

  ```
  getNanoRouteMode
  ```

- The following command displays the current settings for all `setNanoRouteMode` parameters in Tcl list format only:

  ```
  getNanoRouteMode -quiet
  ```

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*

getNetWireLength

```
getNetWireLength
     {netName | -inFile fileName.in}
     [-outFile fileName.out]
```

Reports the length, in microns, of the specified net. You can also use this command to find the length of a prerouted net.

You can use the `getNetWireLength` command after importing a design.

Parameters

`-inFile` *`fileName`*`.in`

Reports the lengths of all the nets listed in the specified file. Wildcards can be used when specifying net names.

`netName`

Specifies the name of the net for which to report the length. Wildcards can be used when specifying net names.

`-outFile` *`fileName`*`.out`

Saves the generated report of net lengths to the specified file.

Examples

- The following command reports the length of the net `n_5101`:

  ```
  getNetWireLength n_5101
  ```

  The software reports:

  ```
  n_5101 : 28.29
  ```

- The following commands reports the lengths of all nets with names that start with `n_`:

  ```
  getNetWireLength n_*
  ```

  The software reports:

  ```
  n_5101 : 28.29
  n_3419 : 185.86
  n_309 : 0.0
  n_339 : 0.0
  n_364 : 0.0
  n_377 : 0.0
  n_393 : 0.0
  ```

```
n_397 : 0.0
n_402 : 643.84
n_404 : 0.0
n_3297 : 108.6
n_3302 : 70.86
```

- The following command reprts the lengths of all nets with names that start with n_ to a file named `NetLengths.out`:

```
getNetWireLength n_* -outFile NetLengths.out
```

getTrialRouteMode

```
getTrialRouteMode
     [-autoSkipTracks]
     [-blockageCostMultiple]
     [-detour]
     [-excludePartition]
     [-extendM1PinToM2]
     [-fastRouteForPinAssign]
     [-floorPlanMode]
     [-handleEachPartition]
     [-handleEachPD]
     [-handlePartFixedNets]
     [-handlePartition]
     [-handlePartitionComplex]
     [-handlePD]
     [-handlePDComplex]
     [-handlePreroute]
     [-highEffort]
     [-honorNetRTLayer]
     [-honorPin]
     [-honorRoutingHalo]
     [-honorActiveLogicView]
     [-ignoreAbutted2TermNet]
     [-ignoreBumpNets]
     [-ignoreGuideLayer]
     [-ignoreNetIsSpecial]
     [-ignoreObstruct]
     [-ignorePrefExtraSpace]
     [-ignoreRTBlockage]
     [-intraNets]
     [-keepEndPoints]
     [-keepEndPoints2]
     [-keepExistingRoutes]
     [-keepPreferredLayer]
     [-maxDetourRatio]
     [-maxRouteLayer]
     [-minRouteLayer]
     [-PDAwareSelnet]
     [-pinGuide]
     [-PKS]
     [-printSections]
     [-printWiresOnRTBlk]
     [-printWiresOnRTBlkFile]
     [-printWiresOnRTBlkLong]
     [-printWiresOnRTBlkLongFile]
     [-printWiresOutsideBusguide]
     [-ptnBdryExt]
     [-ptnBdryShr]
     [-quiet]
     [-routeGuide]
```

```
[-routeObs]
[-selMarkedNet]
[-selMarkedNetOnly]
[-selNet]
[-selNetOnly]
[-skipTracks]
[-updateRemainTrks]
[-useM1]
```

Displays the following information about a specified Trial Route mode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the Trial Route mode parameters.

Parameters

`-parameter_names`

Displays information for the specified parameters. You can specify one or more parameters.

See `setTrialRouteMode` for descriptions of the Trial Route parameters you can specify.

`-quiet`

Displays the current settings for the specified parameters in Tcl list format only.

If you specify `-quiet` without any parameters, the software displays the current settings of all `setTrialRouteMode` parameters in Tcl list format.

Examples

- The following command displays the current setting for the `-handlePartitionComplex` parameter:

  ```
  getTrialRouteMode -handlePartitionComplex
  ```

  The software displays the following information:

```
-handlePartitionComplex false            # bool, default=false
false
```

- The following command displays the current setting for the `-printSections` and `-floorPlanMode` parameters:

  ```
  getTrialRouteMode -printSections -floorPlanMode
  ```

  The software displays the following information:

  ```
  -printSections false                    # bool, default=false
  -floorPlanMode false                    # bool, default=false
  {printSections false} {floorPlanMode false}
  ```

- The following command displays the current setting of the `-handlePartitionComplex` parameter in Tcl list format only:

  ```
  getTrialRouteMode -handlePartitionComplex -quiet
  ```

  The software displays the following information:

  ```
  false
  ```

- The following command displays the current settings of the `-printSections` and `-floorPlanMode` parameters in Tcl list format only:

  ```
  getTrialRouteMode -printSections -floorPlanMode -quiet
  ```

  The software displays the following information:

  ```
  {printSections false} {floorPlanMode false}
  ```

globalDetailRoute

```
globalDetailRoute
     [-select]
     [x1 y1 x2 y2]
```

Uses the NanoRoute router to perform both global and detailed routing with one command. When you load a partially routed database, the router completes the routes.

- To perform global routing only, use `globalRoute`. For information, see “globalRoute” on page 1248.
- To perform detailed routing only, use `detailRoute`. For information, see “detailRoute” on page 1231.
- To perform global and detailed routing using standalone NanoRoute, use `globalDetailRouteBatch`. For information, see “globalDetailRouteBatch” on page 1247.

This command supports multiple-CPU processing. For information, see Multiple-CPU Processing Commands and Accelerating the Design Process by Using Multiple-CPU Processing.

Use this command after completing the following steps:

1. Placing the design.
2. Routing special nets.
3. Setting net attributes with `setAttribute`, and NanoRoute run-time options with `setNanoRouteMode`.

   To use NanoRoute current values for attributes, and default values for options, this step is optional.

   For information on `setAttribute`, see “setAttribute” on page 1271. For information on `setNanoRouteMode`, see “setNanoRouteMode” on page 1280.

Parameters

`-select`	Routes selected nets. To select a net, use the `selectNet` command or click the net in the design display window. For information on `selectNet`, see “selectNet” on page 556.

Tip

You can also specify `setNanoRouteMode routeSelectedNetOnly true` to route the selected nets. For information, see "setNanoRouteMode" on page 1280.

x1 y1 x2 y2

Specifies the x and y coordinates of the bounding box for detailed routing. The entire design is globally routed.

Note: If you have not already run detailed routing on the entire design, you cannot run detailed routing on a specified area.

Example

The following commands run global and detailed routing, save the design as `droute`, and generate a routed DEF file named `droute.def`.

```
globalDetailRoute
saveDesign droute
defOut -floorplan -placement -routing droute.def
```

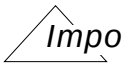
Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - "Routing Phases"

globalDetailRouteBatch

`globalDetailRouteBatch`

Runs global and detailed routing with the NanoRoute router in standalone mode from the Encounter software. The Encounter software passes all current net attribute and option settings, and the selected set of nets, to the router.



Important

Command `globalDetailRouteBatch` is obsolete and has been replaced by `globalDetailRoute`.

Use this command after completing the following steps:

1. Placing the design.
2. Routing special nets.
3. Setting net attributes with `setAttribute`, and NanoRoute run-time options with `setNanoRouteMode`. To use current values for attributes and default values for options, this step is optional.

   For information on `setAttribute`, see “setAttribute” on page 1271. For information on `setNanoRouteMode`, see “setNanoRouteMode” on page 1280.

Parameters

None

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - “Routing Phases”

globalRoute

`globalRoute`

Plans the interconnect by breaking the routing portion of the design into rectangles called global routing cells (gcells) and assigning the signal nets to the gcells.

Note: If you are running ECO routing, use `globalDetailRoute`. Your results are not predictable with other routing commands.

Use this command after completing the following steps:

1. Placing the design.
2. Setting net attributes with `setAttribute`, and NanoRoute run-time options with `setNanoRouteMode`. To use current values for attributes and default values for options, this step is optional.

   For information on `setAttribute`, see "setAttribute" on page 1271. For information on `setNanoRouteMode`, see "setNanoRouteMode" on page 1280.

Parameters

None

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - "Global Routing"
- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*

reportCongestArea

```
reportCongestArea
     [-num [all | numOfHotSpot]]
     [-cutOffRatio cutOffRatioOfPeakHotSpot]
     [-cutOffValue cutOffCongestNum]
     [-step num]
     [-cutoffArea cutOffGcellNum]
     [-outfile fileName]
```

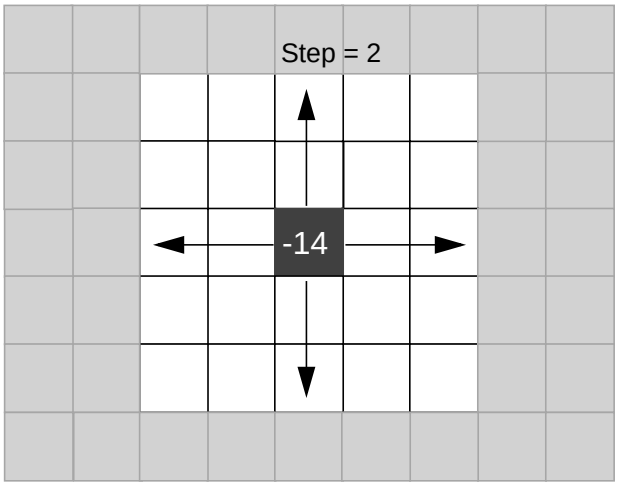
Reports the congestion hot spots in the design.

The Encounter software uses the following algorithm to define the boundaries of hot spots:

1. Finds the most over-congested gcell (that is, the gcell with the largest overflow value) in the design.

   A gcell has overflow if its demand exceeds its supply. Typically, the supply is the number of unobstructed tracks crossing the gcell, and the demand is the number of wires assigned to it.

2. Starting from the over-congested gcell, checks a specified number of gcells (using `-step`) in one direction for congestion. If the gcells have an overflow value that is less than the specified congestion threshold (using `-cutOffRatio` or `-cutOffValue`), the software determines this to be the boundary for that side of the hot spot, and begins checking in a new direction.



3. After the software establishes the boundaries for the first hot spot, it finds the next most over-congested gcell that is not part of the already defined hot spot and begins defining the boundaries for it.

You can use the `reportCongestArea` command after running Trial Route.

Parameters

`-cutoffArea` *`cutOffGcellNum`*

Filters out any hot spots that have an area (specified in gcells) that is less than *`cutOffGcellNum`*.

Default: `0` (Reports all hot spots in the design)

`-cutOffRatio` *`cutOffRatioOfPeakHotSpot`*

Specifies the minimum congestion threshold value a gcell must have to be considered part of the hot spot, as a ratio of the peak overflow value.

The software checks the congestion for the gcell by dividing the overflow value of the most over-congested gcell (that is, the peak overflow value) by the overflow value of the gcell. If the resulting congestion value is less than the specified *`cutOffRatioOfPeakHotSpot`*, the software determines this to be the boundary for that side of the hot spot, and begins checking in a new direction.

Default: `0.01`

`-cutOffValue` *`cutOffCongestNum`*

Specifies the minimum overflow value a gcell must have to be considered part of the hot spot.

Default: `-2`

`-num [all |` *`numOfHotSpot`*`]`

Specifies how many hot spots to report.

Default: `10`

`all` Reports all hot spots in the design.

`numOfHotSpot`

Reports up to the specified number of hot spots in the design.

`-outfile` *`fileName`*

Outputs the hot spot report to the specified file.

`-step` *`num`* Specifies the number of gcells to check for congestion.

Default: `3`

Examples

- The following command outputs a report describing up to10 hot spots in the design that are found by checking three gcells at a time, using an absolute congestion threshold of `-2` and a relative threshold value of `0.01`:

  ```
  reportCongestArea -step 3 -cutoffRatio 0.010000 -cutoffValue -2 -num 10
  ```

  The resulting report contains the following hot spot information for vertical and horizontal congestion:

  - The largest overflow value for the hot spot (`#Peak`)
  - The location of the gcell that contains the largest overflow value (`Location`)
  - The boundary coordinates for the hot spot (`Boundary Box`)
  - The area of the hot spot, in gcells (`Area`)

- The following example shows the congestion hot spot report generated by the previous command:

  ```
  #       Vertical Congestion Area Report
  #Peak   Location(X,Y)      Boundary Box                Area(#Gcell)
  -36      (1791.7,  376.4)    (1778,  332) (1792,  786)    (6x123)
  -23      ( 607.2,  284.1)    ( 580,  251) ( 607,  306)    (12x15)
  -20      (1791.7, 2236.1)    (1778, 2214) (1792, 2247)    (6x9)
  -18      ( 607.2,  125.5)    ( 593,   92) ( 607,  148)    (6x15)
  -15      ( 637.1,  380.1)    ( 630,  347) ( 720,  435)    (39x24)
  -14      (1791.7,  859.8)    (1778,  860) (1792,  982)    (6x33)

  #       Horizontal Congestion Area Report
  #Peak   Location(X,Y)      Boundary Box                Area(#Gcell)
  -92      (1777.9, 114.4)    (1750, 4) (1854, 114)        (45x30)
  -60      ( 623.3, 114.4)    ( 472, 4) ( 816, 125)        (150x33)
  ```

- The following line from the example shows that the largest overflow value for one hot spot is `-36`, the gcell that contains the overflow value is located at `1791.7 376.4`, the coordinates for the hot spot boundary are (`1778, 332`) (`1792, 786`), and the hot spot is 6 gcells by 123 gcells in size:

  ```
  -36      (1791.7,  376.4)    (1778,  332) (1792,  786)    (6x123)
  ```

reportRoute

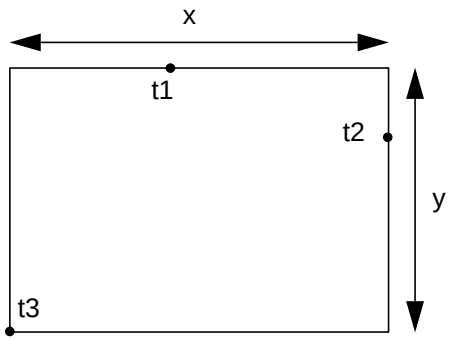
`reportRoute`

Reports routing statistics for signal nets, including the following information:

- Number of objects, such as cells, nets, and pins
- Number and percentage of nets with one, two, three, four, or more terminals
- Names and numbers of primitives used
- Net length and connection length statistics

  These values are computed using the net bounding box half-perimeter.

  The following figure shows a net bounding box for a net that must be connected to three terminals, t1, t2, and t3. The net bounding box is the perimeter of the smallest rectangle that can be formed that has sides passing through the three terminals. The half-perimeter is the length of x plus the length of y.



- Total wire length and number of vias, per layer

  These are the actual wire length and number of vias on each layer. The actual wire length might be different from the net bounding box half-perimeter if the net detours.

  The layer statistics are in the last paragraph of the report.

Note: Use the `reportWire` command to generate a more detailed wire statistics report. For information, see reportWire on page 1258. Use the `reportSpecialRoute` command to generate a report for special nets.

You can use the `reportRoute` command after routing with any of the routing commands in the Encounter software.

Example Report

In the example report the terms `long connection`, `nrCon`, `and total length` are used.

- `long net` is a net that is longer than the last net listed in the report. In the report that follows, it is any net longer than `1.82e+06`.
- `nrCon` is the total number of I/O connections in the nets with long connections.
- `total length` is the total wire length.

```
*** Statistics for net list netlist ***
Number of cells      = 149
Number of nets       = 148
Number of tri-nets   = 0
Number of degen nets = 1
Number of pins       = 466
Number of i/os       = 13

Number of nets with    1 terms = 1 (0.7%)
Number of nets with    2 terms = 120 (81.1%)
Number of nets with    3 terms = 18 (12.2%)
Number of nets with    4 terms = 1 (0.7%)
Number of nets with >=10 terms = 8 (5.4%)

*** 10 Primitives used:
Primitive PVSS1DGZ (4 insts)
Primitive PVDD1DGZ (4 insts)
Primitive PDO04CDG (6 insts)
Primitive PDIDGZ (7 insts)
Primitive BUFX2 (5 insts)
Primitive INVXL (3 insts)
Primitive MXI2XL (20 insts)
Primitive NAND3XL (5 insts)
Primitive NAND4XL (15 insts)
Primitive DFFXL (80 insts)
************
*** Net length and connection length statistics (cell netlist) ***

Total net length = 8.323e+04 (3.543e+04 4.780e+04)

Avg net length = 5.624e+02 (sigma = 1.136e+03)
Sqrt of avg square net length = 1.268e+03

Avg connection length = 2.609e+02 (sigma = 5.590e+02)
Sqrt of avg square connection length = 6.169e+02

Net and connection length distribution:

[length   range   ]      #net #connection
[0.00e+00 9.20e+00]:       32         32
[9.20e+00 1.84e+01]:       25         26
[1.84e+01 2.76e+01]:       18         19
[2.76e+01 3.68e+01]:       10         11
[3.68e+01 4.60e+01]:        7          7
[4.60e+01 5.52e+01]:        4          4
```

```
[5.52e+01 6.44e+01]:        1        2
[6.44e+01 7.36e+01]:        1        1
[8.28e+01 9.20e+01]:        3        2
[9.20e+01 1.01e+02]:        0        1
[1.01e+02 1.10e+02]:        1        1
[1.10e+02 1.20e+02]:        1        1
[1.20e+02 1.29e+02]:        2        2
[1.29e+02 1.38e+02]:        2        3
[1.75e+02 1.84e+02]:        2        1
[1.93e+02 2.02e+02]:        1        1
[2.39e+02 2.48e+02]:        1        0
[2.58e+02 2.67e+02]:        1        3
[2.67e+02 2.76e+02]:        1        2
[2.76e+02 2.85e+02]:        0        1
[2.85e+02 2.94e+02]:        0        1
[3.31e+02 3.40e+02]:        0        1
[3.40e+02 3.50e+02]:        0        1
[3.86e+02 3.96e+02]:        0        1
[3.96e+02 4.05e+02]:        1        1
[4.05e+02 4.14e+02]:        1        1
[4.60e+02 4.69e+02]:        1        1
[5.24e+02 5.34e+02]:        1        1
[5.43e+02 5.52e+02]:        2        1
[6.35e+02 6.44e+02]:        1        1
[6.72e+02 6.81e+02]:        1        0
[6.81e+02 6.90e+02]:        1        0
[6.99e+02 7.08e+02]:        1        1
[7.18e+02 7.27e+02]:        1        1
[7.54e+02 7.64e+02]:        1        1
[1.82e+06 infinity]:       23       15

Net input3c nrTerm=3 conLen=2.37e+06 len=(545880 4203530)
  Inst ibscell3/i_4 loc=(1036300 716510)
  Inst ibscell3/i_1 loc=(970060 716510)
  Inst ipad3 loc=(491000 4915650)
Net scanout3 nrTerm=3 conLen=3.87e+06 len=(3741820 3997350)
  Inst ibscell4/i_1 loc=(4601300 4708270)
  Inst ibscell3/i_3 loc=(1012380 714870)
  Inst ibscell3/i_2 loc=(995820 716510)
Net input4c nrTerm=3 conLen=2.26e+06 len=(294750 4217650)
  Inst ibscell4/i_4 loc=(4584740 4707450)
  Inst ibscell4/i_1 loc=(4605900 4707450)
  Inst ipad4 loc=(4878350 491000)
Net input5c nrTerm=3 conLen=2.32e+06 len=(431980 4199680)
  Inst ibscell5/i_4 loc=(4501020 4707450)
  Inst ibscell5/i_1 loc=(4532300 4707450)
  Inst ipad5 loc=(4933000 508350)
Net output1c nrTerm=2 conLen=3.78e+06 len=(449750 3335110)
  Inst ipad6 loc=(4933000 1375380)
  Inst ibscell6/i_4 loc=(4483540 4709910)
Net scanout8 nrTerm=3 conLen=3.25e+06 len=(2497770 4002270)
  Inst ibscell9/i_1 loc=(2172500 715690)
  Inst ibscell8/i_3 loc=(3177140 4709090)
  Inst ibscell8/i_2 loc=(3225900 4710730)
Net output4c nrTerm=2 conLen=5.37e+06 len=(1145120 4220210)
  Inst ipad9 loc=(552620 4933000)
  Inst ibscell9/i_4 loc=(1696860 714050)
Net output5c nrTerm=2 conLen=5.14e+06 len=(917670 4221910)
  Inst ipad10 loc=(1645120 4933000)
  Inst ibscell10/i_4 loc=(1647180 714050)
Net doutm1_12 nrTerm=2 conLen=6.82e+06 len=(2825960 3998370)
```

```
  Inst i1_15 loc=(4175340 4708270)
  Inst i1_12 loc=(1902020 713230)
Net doutm2_9 nrTerm=2 conLen=6.41e+06 len=(2418860 3995400)
  Inst i2_15 loc=(3674860 4706630)
  Inst i2_9 loc=(2307740 713230)
Net doutm2_10 nrTerm=2 conLen=6.46e+06 len=(2438640 4018870)
  Inst i2_15 loc=(3673940 4707450)
  Inst i2_10 loc=(2292100 713230)
Net doutm2_12 nrTerm=2 conLen=6.47e+06 len=(2454740 4018870)
  Inst i2_15 loc=(3672100 4709090)
  Inst i2_12 loc=(2274620 713230)
Net doutm3_9 nrTerm=2 conLen=6.16e+06 len=(2142400 4018050)
  Inst i3_15 loc=(2982100 4706630)
  Inst i3_9 loc=(2596620 713230)
Net doutm3_10 nrTerm=2 conLen=6.17e+06 len=(2172340 3999190)
  Inst i3_15 loc=(2981180 4706630)
  Inst i3_10 loc=(2561660 713230)
Net doutm3_11 nrTerm=2 conLen=6.17e+06 len=(2152980 4017230)
  Inst i3_15 loc=(2980260 4708270)
  Inst i3_11 loc=(2579140 713230)

Total number of long connections = 35
Io: nrCon=0
HInst topLevel: nrCon=20
HInst ibscell3: nrCon=4
HInst ibscell4: nrCon=3
HInst ibscell5: nrCon=2
HInst ibscell6: nrCon=1
HInst ibscell8: nrCon=2
HInst ibscell9: nrCon=2
HInst ibscell10: nrCon=1


Total length: 9.512e+04um, number of vias: 1214
M1(H) length: 8.847e+02um, number of vias: 511
M2(V) length: 1.185e+04um, number of vias: 590
M3(H) length: 4.210e+04um, number of vias: 91
M4(V) length: 3.641e+04um, number of vias: 22
M5(H) length: 3.885e+03um
*** Report route (0:00:00.0 235.1M) ***
```

reportShield

```
reportShield
     [-selected]
     [-include_layer]
     [-verbose]
     [-out_file fileName]
     [-help]
```

Reports shield coverage for nets routed by the NanoRoute router.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reportShield` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man reportShield`.
`-include_layer`	Reports the shielding ratio per layer per net in addition to the ratio for the whole net..
`-out_file` *`fileName`*	Specifies the report file name. *Default:* *`designName`*`_shield.rpt`
`-selected`	Reports statistics for selected nets only.
`-verbose`	Includes a list of nets that are skipped, which includes the following nets: ■ `FIXED` nets ■ `SPECIAL` nets ■ Nets with the `skip_routing` attribute ■ Nets with opens

Examples

The following command generates a report called `myDesign_shield.rpt` for selected nets. The report includes information for the net on a per-layer basis.

```
reportShield -include_layer -selected -out_file myDesign_shield.rpt
```

Related Topics

- "Using the NanoRoute Router" chapter in the *Encounter User Guide*
  - Interpreting the Shielding Report

reportWire

`reportWire [-detail] [`*`fileName`*`] [`*`ratio`*`]`

Creates a report file that contains wire statistics for signal nets with a real wire length or minimum wire length that is greater than the specified ratio.

Ratio (*r*) is calculated as *r* = (*total net routing length*) / (*half-perimeter of net bounding box*).

The wire information reported includes the net name, the wire ratio, the real and minimum wire lengths, the number of vias, and the number of fanins and fanouts on the net.

Note: Use the `reportRoute` command to generate a layer-by-layer wire statistics report. For information, see reportRoute on page 1252. Use the `reportSpecialRoute` command to generate a report for special nets.

Use the `reportWire` command after routing with any of the routing commands in the Encounter software.

Parameters

`-detail`	Includes the following additional information in the report: the wire length and number of vias for each layer, and a summary of the total wire length and via number for the design.
`fileName`	Creates a report file using the specified name. *Default:* Creates a report file named `topcell.wirerpt`
`ratio`	Reports wire statistics for wires with lengths greater than the specified ratio. If you do not specify a ratio, the `reportWire` command calculates wire information for wires with a ratio greater than `1.0`. *Default:* `1.0` **Note:** There are many situations in which the ratio can be less than `1.0`. To ensure that the software creates a report on all nets, specify a ratio value of `0`.

Examples

- The following command writes the wire statistics for all nets with wire lengths greater than the 1.0 ratio:

  ```
  reportWire 1.0
  ```

The following example shows a section of the wire information reported:

```
ratio     wire_length   manhattan    vias  fanIn  fanOut net_name
                        length
------------------------------------------------------------------------
1.08 Total  177.090     164.16 Total  4     1      1   dOut2[1]
1.12 Total   79.185      70.94 Total  3     1      1   dOut2[0]
1.04 Total  179.960     173.46 Total  7     1      4   dIna2[1]
1.00 Total  178.680     178.02 Total  8     1      4   dIna2[0]
1.02 Total  203.420     199.76 Total  10    1      4   dInb2[1]
1.25 Total  222.420     178.30 Total  12    1      4   dInb2[0]
1.18 Total  185.645     157.32 Total  11    1      4   s2[1]
1.09 Total  456.740     155.76 Total  10    1      4   s2[0]
.
.
.
```

- The following command writes the detailed wire statistics for all nets with wire lengths greater than the 1.0 ratio to the `detail.wirerpt` file. This report includes the wire length and number of vias for each layer, and a summary of the total wire length and via number for the design:

  ```
  reportWire -detail detail.wirerpt 1.0
  ```

  The following example shows a section of the wire information and the summary from the end of the report:

  ```
  ratio     wire_length   manhattan    vias  fanIn  fanOut net_name
                          length
  ------------------------------------------------------------------------
  1.08 Total  177.090     164.16 Total  4     1      1   dOut2[1]
     M1         7.650       V01   0
     M2       103.440       V12   2
     M3        66.000       V23   2
     M4         0.000       V34   0
     M5         0.000       V45   0
     M6         0.000       V56   0
  1.12 Total   79.185      70.94 Total  3     1      1   dOut2[0]
     M1         8.245       V01   0
     M2        61.040       V12   1
     M3         9.900       V23   2
     M4         0.000       V34   0
     M5         0.000       V45   0
  .
  .
  ```

```
.
### Summary Report ###

Total M1 Wire Length =      33.99
Total M2 Wire Length =    1869.13
Total M3 Wire Length =    1885.62
Total M4 Wire Length =      67.76
Total M5 Wire Length =       0.00
Total M6 Wire Length =       0.00
---------------------------------
Total Wire Length    =    3856.51

Total V01 Number     =          0
Total V12 Number     =        272
Total V23 Number     =        242
Total V34 Number     =          3
Total V45 Number     =          0
Total V56 Number     =          0
---------------------------------
Total Via Number     =        517
```

restoreRoute

`restoreRoute` *`fileName`*

Loads routing data into the design.

You can use the `restoreRoute` command after loading the placement file.

Parameters

`fileName`	Specifies the filename of the routing data to load.

routeDesign

```
routeDesign
     [-global | -detail | -globalDetail | -viaOpt | -wireOpt] [-noPlacementCheck]
```

Runs routing or postroute via or wire optimization using the NanoRoute router. If you specify this command without any arguments, it runs global and detailed routing.

`routeDesign` has the following characteristics:

- It supports multiple-CPU processing. For information, see Multiple-CPU Processing Commands and Accelerating the Design Process by Using Multiple-CPU Processing.
- It checks for conflicts in option settings and issues warning messages when it finds problems. For example, trying to fix postroute lithography problems and optimize vias concurrently can cause conflicts. If `routeDesign` detects requests for both types of operation, it issues a warning, turns off via optimization, and proceeds with fixing lithography problems.
- It honors all `setNanoRouteMode` settings that do not have conflicts
- It is timing and SI driven by default
  - To turn off timing-driven routing, run the following command before running `routeDesign`:

    ```
    setNanoRouteMode -routeWithTimingDriven false
    ```

  - To turn off SI-driven routing, run the following command before running `routeDesign`:

    ```
    setNanoRouteMode -routeWithSIDriven false
    ```

    If you run either (or both) of these commands, the software stores the mode settings in the `.mode` file. If you restore the design and run `routeDesign` again, it honors the these settings. For example, if you type the following commands, the next time you restore the design and run `routeDesign`, it routes the design in non-timing-driven mode:

    ```
    setNanoRouteMode -routeWithTimingDriven false
    routeDesign -detail
    saveDesign
    ```

- It updates the timing file automatically after all the nets are detail routed in timing-driven mode. Updating the file ensures that critical nets are not touched during postroute multiple-cut via swap, wire spreading and wire widening. When determining whether all the nets are detail routed, the router skips pins without physical ports.
- It runs a placement check prior to routing and displays a warning message if the placement is not clean.

To turn off the placement check, specify the following parameter: `-noPlacementCheck`

- It routes clock nets first if you specify the following `setNanoRouteMode` parameter:

  `-routeDesignRouteClockNetsFirst`

See the following commands for more information:

- detailRoute on page 1231
- globalDetailRoute on page 1245
- globalRoute on page 1248
- setNanoRouteMode on page 1280

Parameters

`-detail`	Runs timing-driven and SI-driven detailed routing.
`-global`	Runs timing-driven and SI-driven global routing.
`-globalDetail`	Runs timing-driven and SI-driven global and detailed routing. `-globalDetail` is the default value for this command.
`-noPlacementCheck`	Turns off placement checking. By default, `routeDesign` runs a placement check before routing and displays a warning message if the placement is not clean. For information on the types of checks and the violation report, see `checkPlace`.
`-viaOpt`	Optimizes vias after routing. *Caution* ***Exercise caution with the use of this parameter, as the router might create violations during postroute via optimization. It runs a search and repair step after optimization if violations occur.***
`-wireOpt`	Optimizes wires after routing.

Examples

Routing

To run timing- and SI-driven global and detailed routing, type the following command:

```
routeDesign
```

To run timing- and SI-driven global routing only, type the following command:

```
routeDesign -global
```

To run non-timing-driven global routing, type the following commands:

```
setNanoRouteMode -routeWithTimingDriven false
routeDesign -global
```

To run timing- and SI-driven detailed routing, type the following command:

```
routeDesign -detail
```

To run non-timing-driven detailed routing, type the following commands:

```
setNanoRouteMode -routeWithTimingDriven false
routeDesign -detail
```

To run timing- and SI-driven global and detailed routing, type the following commands:

```
routeDesign -globalDetail
```

To run non-timing-driven global and detailed routing, type the following commands:

```
setNanoRouteMode -routeWithTimingDriven false
routeDesign -globalDetail
```

Postroute Yield Optimization

To run postroute via reduction, type the following commands:

```
setNanoRouteMode -drouteMinSlackForWireOptimization slack
setNanoRouteMode -droutePostRouteMinimizeViaCount true
routeDesign -viaOpt
```

> **Note:** When you run these commands, the software replaces all multiple-cut vias with single-cut vias. Use these commands only if no concurrent via optimization was done.

To run postroute single-cut via to multiple-cut via swap, type the following commands:

```
setNanoRouteMode -drouteMinSlackForWireOptimization slack
setNanoRouteMode -droutePostRouteSwapVia multiCut
routeDesign -viaOpt
```

To run non-timing-driven postroute single-cut via to multiple-cut via swap, type the following commands:

```
setNanoRouteMode -routeWithTimingDriven false
setNanoRouteMode -droutePostRouteSwapVia multiCut
routeDesign -viaOpt
```

To run postroute multiple-cut via to single-cut via swap, type the following commands:

```
setNanoRouteMode -drouteMinSlackForWireOptimization slack
setNanoRouteMode -droutePostRouteSwapVia singleCut
routeDesign -viaOpt
```

To run non-timing driven postroute multiple-cut via to single-cut via swap, type the following commands:

```
setNanoRouteMode -routeWithTimingDriven false
setNanoRouteMode -droutePostRouteSwapVia singleCut
routeDesign -viaOpt
```

To run postroute wire widening, type the following commands:

```
setNanoRouteMode -drouteMinSlackForWireOptimization slack
setNanoRouteMode -droutePostRouteWidenWireRule ruleName
setNanoRouteMode -droutePostRouteWidenWire widen
routeDesign -wireOpt
```

To run non-timing driven postroute wire widening, type the following commands:

```
setNanoRouteMode -droutePostRouteWidenWireRule ruleName
setNanoRouteMode -routeWithTimingDriven false
setNanoRouteMode -droutePostRouteWidenWire widen
routeDesign -wireOpt
```

To run postroute wire unwidening, type the following commands:

```
setNanoRouteMode -drouteMinSlackForWireOptimization slack
setNanoRouteMode -droutePostRouteWidenWireRule ruleName
setNanoRouteMode -droutePostRouteWidenWire shrink
routeDesign -wireOpt
```

To run non-timing-driven postroute wire unwidening, type the following commands:

```
setNanoRouteMode -droutePostRouteWidenWireRule ruleName
setNanoRouteMode -routeWithTimingDriven false
setNanoRouteMode -droutePostRouteWidenWire shrink
routeDesign -wireOpt
```

To run postroute wire spreading, type the following commands:

```
setNanoRouteMode -drouteMinSlackForWireOptimization slack
setNanoRouteMode -droutePostRouteSpreadWire true
routeDesign -wireOpt
```

To run non-timing-driven postroute wire spreading, type the following commands:

```
setNanoRouteMode -routeWithTimingDriven false
setNanoRouteMode -droutePostRouteSpreadWire true
routeDesign -wireOpt
```

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - "Using the routeDesign Supercommand"
  - "Postroute Via Optimization"
- Accelerating the Design Process by Using Multiple CPU Processing chapter in the *Encounter User Guide*
- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Partitions in the *Encounter Hierarchical Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Top-Level in the *Encounter Hierarchical Implementation Flow Guide*

runCCAR

```
runCCAR
     [-interactive]
     -nets {list_of_nets}
     -doFile do_file_name
     -lib library
     -cell cell
     -view view
     -refLibs reference_libraries
     [-netsInArea {llx lly urx ury}]
```

Places a set of nets into the chip assembly router environment, where you can use the chip assembly router's editing features.

Parameters

`-cell` *`cell`*
Specifies the cell to import into the chip assembly router environment.
Default: The top cell in the design

`-doFile` *`do_file_name`*
Specifies the chip assembly router `.do` file.

Include the following statement in the `.do` file before any routing commands:

```
set rotate_via off
```

Include the following statement in the `.do` file after all routing commands:

```
protect all wires
```

`-interactive`
Specifies that the chip assembly router interface is displayed with the design to enable interactive editing. If you do not specify this parameter, this command runs in batch mode.

`-lib` *`library`*
Specifies the design library to use. The software creates the library if it does not already exist.
Default: `soce2ccar`

`-nets {`*`list_of_nets`*`}`

Specifies the nets to import. Enclose the list of net names in curly braces.

Note: The `-nets` parameter is optional if you use the `-netsInArea` parameter and required if you do not use the `-netsInArea` parameter.

`-netsInArea {`*`llx lly urx ury`*`}`

Specifies the coordinates of the area within the design from which nets are passed to the chip assembly router environment. If a net is partially within the specified area, the entire net is imported into the chip assembly router environment. Enclose the coordinate list in curly braces.

`-refLibs` *`reference_libraries`*

Specifies the libraries that contain the technology information in Open Access format. You can create these libraries using the LEF to OpenAccess translator. For information on the translator, see OpenAccess documentation.

`-view` *`view`*

Specifies a view name for the design you are importing into the chip assembly router environment.
Default: `layout`

saveRoute

```
saveRoute fileName
```

Saves the routing information to a file.

You can use the `saveRoute` command after running Trial Route.

Parameters

`fileName`	Specifies the name of the file in which to save the routing data. Routing data can be saved in uncompressed or compressed (`.gz`) files.

Examples

- The following command saves the routing data to the file `mydesign.route`:

  ```
  saveRoute mydesign.route
  ```

saveRouteGuide

```
saveRouteGuide
     [-rguide routeGuideFileName]
     [-selNetFile selNetFileName]
```

Saves the routing information to a route guide format file. Routing guide files help control the routing topology by creating regions in which Trial Route can create wires.

You can use the `saveRouteGuide` command after running Trial Route or after restoring routing data.

Parameters

`-rguide` *`routeGuideFileName`*

Specifies the route guide file to which to save the routing information.

`-selNetFile` *`selNetFileName`*

Specifies a file of hierarchical net names for which to save route information. Routing information is saved only for these nets.

Examples

- The following command writes the route information in route guide format to the `cts_3.guide` file:

  ```
  saveRouteGuide -guide cts_3.guide
  ```

setAttribute

```
setAttribute
     {-net netName | -cell_pin cell_name:pin_name}{[-weight integer]}
     {[-avoid_detour {true | false}]
      [-bottom_preferred_routing_layer layerNumber]
      [-cell_name cell_name:pin_name]
      [-non_default_rule ruleName]
      [-pattern {steiner | trunk}]
      [-preferred_extra_space integer]
      [-preferred_routing_layer_effort {low | medium | high}]
      [-shield_net specialNetName]
      [-si_post_route_fix {true | false}]
      [-skip_antenna_fix {true | false}]
      [-skip_routing {true | false}]
      [-top_preferred_routing_layer layerNumber]}
```

Attaches attributes to nets. Attaching the attributes allows the NanoRoute routing commands (`globalRoute`, `detailRoute`, `globalDetailRoute`, `globalDetailRouteBatch`, `routeDesign`, and `setNanoRouteMode`), and the `trialRoute` command to route the nets following specific requirements. Attributes are persistent; that is, throughout the routing process, from global routing to optimization, the routers honor the attributes. The attributes are saved in the Encounter database. If you save the database and exit, the attributes remain attached to the nets when you reimport the database.

Note: Attributes are not saved in the DEF file.

To see the current attribute settings, use `getAttribute`. For information, see "getAttribute" on page 1236.


Tip

To revert to the default value of an attribute, specify `default` for the attribute.

Parameters

`-cell_pin cell_name:pin_name`

Specifies attributes for all the nets connected to pins of instances having the same master cells. You can use `:` or `/` as delimiter when you specify the cell name and pin name.

Note: If the specified `setAttribute` commands conflict, then the last setting takes precedence over all the previous `setAttribute` settings.

`-net` *`netName`*

Specifies the net for which to set attributes. Do not enter more than one net name.

- To specify more than one net, use * and ? wildcard characters.
- To specify nondefault rule nets, use `@RULE`.
- To specify clock nets, use `@CLOCK`. (To select clock nets use `selectNet -allDefClock`.)
- To specify prerouted nets, use `@PREROUTED`.

Remove an attribute from a net by specifying `none`. For example, to remove the `-shield_net` attribute from `net_1`, specify the following command:

```
setAttribute -net net_1 -shield_net none
```

Specify one or more of the following attributes:

`-avoid_detour {true | false}`

Avoids detours of roughly more than a few gcell grids on the specified nets. This attribute affects global routing only.

Cadence recommends that you use caution with this attribute, as it adds congestion to the design.

`-bottom_preferred_routing_layer` *`layerNumber`*

Specifies the preferred lowest routing layer. This attribute is a soft limit; that is, NanoRoute might use a layer below the specified layer if necessary to complete routing.
Range: `0-15`

Note: If the top routing layer is equal to or greater than 4 and no bottom preferred routing layer is specified for clock nets, the router automatically sets the bottom preferred routing layer for clock nets to 3, to prevent the clock nets from blocking other nets.

`-non_default_rule` *`ruleName`*

Identifies the nondefault routing rule to use with the specified net. Use this attribute for critical nets, such as clock nets, that might need to have wider wires or wider spacing than other nets.

The NanoRoute router treats nondefault rule spacing as a soft attribute; that is, when routing resources are available, it honors the nondefault rule. If the area is too congested, and resources are not available, the router might not honor nondefault spacing rules. You can override this behavior by specifying `setNanoRouteMode -strictlyHonorNonDefaultRule true`. When you specify this parameter, the router uses the nondefault spacing rules.

You define nondefault routing rules in the `NONDEFAULTRULE` statement of the LEF file.

`-pattern {steiner | trunk}`

`steiner`

Routes all pins with a single steiner tree.

`trunk`

Routes pins to the nearest special net of the same name. Creates multiple steiner trees and connects them to the trunk.

`-preferred_extra_space` *`integer`*

Gives additional pitch spacing to the specified net. Use this attribute to give critical nets extra space to reduce coupling. Specify `-preferred_extra_space 1` to give a net `1` extra pitch spacing, compared to other nets. This parameter creates a soft rule and, if the design is congested, might not give the extra space to the net.
Range: `0-3`

Note: If you specify this attribute, Trial Route also gives additional space to the specified nets.

`-preferred_routing_layer_effort {low | medium | high}`

Determines how flexible the NanoRoute router is when setting layer limits for routing specified nets. Use this attribute with the `-top_preferred_routing_layer` and `-bottom_preferred_routing_layer` attributes. Specify `medium` or `high` to make NanoRoute less flexible (that is, to make NanoRoute obey the preferred routing layer range more strictly).
Default: `low` (more flexible)

`-shield_net` *`specialNetName`*

Specifies a special net to use as a shield for a critical or high-speed net. Typically, you route shielded nets before routing other nets. You can specify one or two shield nets, however, you cannot control which side is used by the special net that is routed on the same layer as the signal routing. The shield terminates near the pin. Shielding information is saved in the DEF file.

Note: The router does not support shielding nets that are defined in the `SPECIALNET` section of the DEF file. This limitation extends to nets that are defined partly in the `SPECIALNET` section and partly in the regular `NET` section of the file, such as some clock nets. If you set the shielding attribute on these nets, the router automatically sets the `-skip_routing` attribute on them as well. As a result, the nets are not modified by the router and, if they are not completely routed already, the router does not try to finish the connections. The router does not delete the shielding wires.

`-si_post_route_fix {true | false}`

Repairs crosstalk violations on specified nets. If you are using a third-party tool for noise analysis, use this attribute in combination with the `-routeWithSiPostRouteFix` parameter after you run noise analysis. For information on `-routeWithSiPostRouteFix`, see “setNanoRouteMode” on page 1280.

`-skip_antenna_fix {true | false}`

Prevents the router from repairing violations on specified nets with process antenna violations. This attribute is useful on nets that will have antenna violations repaired at the next level of hierarchy.

`-skip_routing {true | false}`

Prevents the NanoRoute router from routing or rerouting unrouted or partially routed nets.

When you specify `-skip_routing` for a net, the router treats the net as if you had marked it `+ FIXED` in the DEF file and it becomes an obstruction for other nets.

If the net is partially routed, the router does not complete the routing.

The value you specify for this attribute also applies to Trial Route.

Note: To skip routing on prerouted nets, specify `setAttribute -net @PREROUTED -skip_routing true`.

`-top_preferred_routing_layer` *`layer_number`*

Specifies the preferred highest layer for routing specified signal nets. This attribute is a soft limit; that is, NanoRoute might use a layer above the specified layer if necessary to complete routing. Use this attribute to improve timing by restricting the routing layers available for critical nets, while simultaneously routing other nets on all routing layers.
Range: `0-15`

`-weight` *`integer`*

Specifies a relative weight for routing nets. In each switch box, the NanoRoute router routes nets with the highest weight first, then the next highest weight, and so on. Specify a value higher than 2 to ensure a net is routed before other nets.
Default: 2

Note: A switch box is a collection of global routing cells used during detailed routing. When you specify a net weight and route a net, the router does not route the entire net according to the weight. It routes the nets in each switch box with the highest weight first, then the next highest weight, and so on.

Examples

- The following command sets the net weight for `net1` to 4:

  ```
  setAttribute -net net1 -weight 4
  ```

  Setting the weight to 4 forces the router to route `net1` before it routes nets whose net weight is less than 4.

- The following command sets the net weight for all nets whose net name starts with `net1` to 4:

  ```
  setAttribute -net net1* -weight 4
  ```

- The following command limits routing for `net1` to layers 4 and 5 if possible:

  ```
  setAttribute -net net1 \
       -top_preferred_routing_layer 5\
       -bottom_preferred_routing_layer 4
  ```

- The following command adds two additional grid units around `net1`. The additional grid units do not have any routing.

  ```
  setAttribute -net net1 -preferred_extra_space 2
  ```

- The following command uses the nondefault rule `wide_wire1` for routing `net1`:

  ```
  setAttribute -net net1 -non_default_rule wide_wire1
  ```

- The following command uses the default rule for routing `net1`.

  ```
  setAttribute -net net1 -non_default_rule default
  ```

- The following commands shield `net1` with both power and ground wires, and shield `net2` with a ground wire:

```
setAttribute -net net1 -shield_net vdd \
     -shield_net vss
setAttribute -net2 -shield_net vss
globalDetailRoute
```

- The following commands show how to shield two nets (do not shield more than one net with the same command):

```
setAttribute -net net1 -shield_net abc_gnd
setAttribute -net net2 -shield_net abc_gnd
```

- The following commands run both global and detailed routing and repair signal integrity violations on `net1` during postroute optimization:

```
setAttribute -net net1 -si_post_route_fix true
setNanoRouteMode -routeWithPostRouteFix true
globalDetailRoute
```

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*

setMaxRouteLayer

```
setMaxRouteLayer layerNumber
```

Specifies a routing layer limit for Trial Route, the NanoRoute router, the mixed signal router, standard cell placement, and clock tree synthesis (CTS).

Trial Route, the NanoRoute router, the mixed signal router, standard cell placement, and CTS can each specify individual “local” maximum routing layer limits, which must be less than or equal to the routing layer limit set by `setMaxRouteLayer`, if one is specified. Local routing layer limits that are less than the limit specified with `setMaxRouteLayer`, supersede the `setMaxRouteLayer` limit. If a local routing layer limit is greater than the limit specified with `setMaxRouteLayer`, the local limit is automatically reset to be equal to `setMaxRouteLayer` limit.

To print the current maximum routing layer value set for a specific feature, use one of the following commands:

- `getPlaceMode`
- `getTrialRouteMode`
- `getCTSMode`
- `getNanoRouteMode`

You can use the `setMaxRouteLayer` command at any time during the Encounter session.

Parameters

`layerNumber`	Limits routing to layers at and below the specified layer.

Examples

- The following command sets the routing layer limit to layer *metal4*. This limits routing to layer *metal4* and below.

  ```
  setMaxRouteLayer 4
  ```

- If you specify the following `setTrialRouteMode` command:

  ```
  setTrialRouteMode -maxRouteLayer 5
  ```

  Trial Route automatically resets the `-maxRouteLayer` parameter to `4` because the value is greater than the `setMaxRouteLayer` limit. It also issues a warning that the maximum routing layer cannot be greater than the value set by `setMaxRouteLayer`.

setNanoRouteMode

```
setNanoRouteMode
     [-help]
     [-reset]
     [-dbReportWireExtraction fileName]
     [-dbReportWireExtractionEcoOnly {true | false}]
     [-dbSkipAnalog {true | false}]
     [-drouteAntennaEcoListFile fileName]
     [-drouteAutoCreateShield {true | false}]
     [-drouteAutoStop {true | false}]
     [-drouteCheckMinstepOnTopLevelPin {true | false}]
     [-drouteElapsedTimeLimit minutes]
     [-drouteEndIteration passNumber]
     [-drouteFixAntenna {true | false}]
     [-drouteMinLengthForWireSpreading length_in_microns]
     [-drouteMinLengthForWireWidening length_in_microns]
     [-drouteMinSlackForWireOptimization time_in_nanoseconds}]
     [-drouteNoTaperInLayers "bottomLayerNum:topLayerNum" ]
     [-drouteNoTaperOnOutputPin {true | false}]
     [-drouteOnGridOnly {none | via | all}]
     [-droutePostRouteLithoRepair {true | false}]
     [-droutePostRouteMinimizeViaCount {true | false}]
     [-droutePostRouteSpreadWire {true | false}]
     [-droutePostRouteSwapVia {multiCut | singleCut | none}]
     [-droutePostRouteWidenWire {widen | shrink | none}]
     [-droutePostRouteWidenWireRule ruleName]
     [-drouteSearchAndRepair {true | false}]
     [-drouteStartIteration passNumber]
     [-drouteUseBiggerOverhangViaFirst {true | false}]
     [-drouteUseMultiCutViaEffort {low | medium | high}]
     [-envAdvancedIntegration {true | false}]
     [-envDontUseLicsForThreadings {"[nano][nanor][nedbs][soce]
         [socegps][socegxl]"}
     [-envNumberFailLimit integer]
     [-envNumberProcessor numberProcessors]
     [-envNumberWarningLimit integer]
     [-envSuperthreading
          {"RSH {machineName number_CPUs nanoroute_executable} ..." |
           "LSF {number_CPUs nanoroute_executable [bsub_options]} ..." |
           "SGE {number_CPUs nanoroute_executable [qsub_options]} ..." |
           "SSH {machineName number_CPUs nanoroute_executable} ..." |
           "<option_configuration_file"}]
     [-routeAllowPowerGroundPin {true | false}]
     [-routeAntennaCellName {"cellName" | "list_of_cellNames"}]
     [-routeAntennaPinLimit numberPins]
     [-routeAutoGgrid {true | false}]
     [-routeBottomRoutingLayer layerNumber]
     [-routeConcurrentMinimizeViaCountEffort {low | medium | high}]
     [-routeDeferredShield {true | false}]
     [-routeDeleteAntennaReroute {true | false}]
```

```
[-routeDesignFixClockNets {true | false}]
[-routeDesignNoCheckPlace {true | false}]
[-routeDesignRouteClockNetsFirst {true | false}]
[-routeEcoOnlyInLayers "bottomLayerNum:topLayerNum"]
[-routeExtraViaEnclosure distance]
[-routeFixTopLayerAntenna {true |false}]
[-routeHonorPartition {true | false}]
[-routeHonorPowerDomain {true | false}]
[-routeIgnoreAntennaTopCellPin {true | false}]
[-routeInsertAntennaDiode {true | false}]
[-routeInsertAntennaInVerticalRow {true | false}]
[-routeInsertDiodeForClockNets {true | false}]
[-routeMergeSpecialWire {true | false}]
[-routeMinShieldViaSpan float]
[-routeReverseDirection "(x1 y1 x2 y2) (x3 y3 x4 y4)"]
[-routeSelectedNetOnly {true | false}]
[-routeSiEffort {high | medium | low | min | normal | max}]
[-routeStrictlyHonorNonDefaultRule {true | false}]
[-routeStripeLayerRange "bottomLayerNum:topLayerNum"]
[-routeTdrEffort integer]
[-routeTopRoutingLayer layerNumber]
[-routeTrunkWithClusterTargetSize integer]
[-routeUseBlockageForAutoGgrid {true | false}]
[-routeWithEco {true | false}]
[-routeWithLithoDriven {true | false}]
[-routeWithSiDriven {true | false}]
[-routeWithSiPostRouteFix {true | false}]
[-routeWithTimingDriven {true | false}]
[-routeWithViaInPin {true| false | bottomLayerNum:topLayerNum}]
[-routeWithViaOnlyForStandardCellPin {true | false |
     bottomLayerNum:topLayerNum}]
[-timingEngine {CTE | externalTimingGraph}]
```

Controls certain aspects of how the NanoRoute router routes the design.

Use the getNanoRouteMode command to return the current settings for the setNanoRouteMode command.

The setNanoRouteMode parameters affect the behavior of the following commands:

- detailRoute
- globalRoute
- globalDetailRoute
- globalDetailRouteBatch
- routeDesign

Parameters

`-dbReportWireExtraction` *`fileName`*

Reports wire extraction information in the specified file.
Default: `""` (empty string)

`-dbReportWireExtractionEcoOnly {true | false}`

Reports only ECO nets during wire extraction.
Default: `false`

`-dbSkipAnalog {true |false}`

Skips routing nets or pins marked `+ USE ANALOG` in the DEF file. Specify this parameter before running any routing commands.
Default: `false`

`-drouteAntennaEcoListFile` *`fileName`*

Specifies the file generated during process antenna repair that contains the list of diodes that were inserted. If you insert diodes during more than one detailed routing pass, the NanoRoute router creates a separate file for each pass and gives it a unique name. For example, if you use the default filename, the router creates the following files:

- After the first detailed routing pass:
  `nano_ant_diode.list`
- After the second detailed routing pass:
  `nano_ant_diode.list1`
- After the third detailed routing pass:
  `nano_ant_diode.list2`

Default: `""` (empty string)

`-drouteAutoCreateShield {true | false}`

Creates shield wires after detailed routing.

To delay creating shields, specify `false` for this parameter and use the `createShield` command to add the shield wires later.

Default: `true`

`-drouteAutoStop {true | false}`

Controls whether NanoRoute continues routing if there are many violations. In highly congested designs, NanoRoute stops if the number of violations is too high. Specify `false` to continue routing until there is no improvement.
Default: `true`

`-drouteCheckMinstepOnTopLevelPin {true | false}`

Checks for minimum step violations where a wire connects to a DEF pin shape. Set this parameter to `true` if the DEF pin shapes in the design are not the same width as the wires.
Default: `false`

`-drouteElapsedTimeLimit` *`minutes`*

Specifies a run-time limit for detailed routing. The time-limit is based on the elapsed time (the wall-clock time), not the CPU time. If the router reaches the limit during initial routing iteration, it stops when it finishes the routing iteration and keeps the routing result. If it reaches the limit during search and repair or postroute optimization, it stops immediately and keeps the routing result at that time.
Default: `0` (no time limit)

`-drouteEndIteration` *`passNumber`*

Specifies the last pass in a detailed routing step. Use with `-drouteStartIteration` to divide detailed routing into intermediate steps.

To run postroute optimization and repair remaining violations, including process antenna violations, specify `-drouteEndIteration default`.
Default: 0 (dynamic default setting)

Note: Specifying a higher value than 20 for `-drouteEndIteration` generally does not improve results.

`-drouteFixAntenna {true | false}`

Repairs process antenna violations by jumping layers (antenna stapling). the router repairs process antenna violations if it can do so without creating DRC violations. The router might need to make several passes before repairing all antenna violations.
Default: `true`

`-drouteMinLengthForWireSpreading` *`length_in_microns`*

Specifies the minimum length in microns for a wire segment of a net before a jog is necessary to avoid OPC issues. Spreading wires farther apart (creating jogs in the wires) helps avoid design rule violations, signal integrity problems, and yield loss caused by shorts in the wires.
Default: 2

Note: Turn wire spreading on or off with `-droutePostRouteSpreadWire`.

`-drouteMinLengthForWireWidening` *`length_in_microns`*

Specifies the minimum length in microns for a wire segment of a net before the segment is a candidate for widening. Wires are widened by an amount defined by a nondefault rule. They are not widened unless they meeting the following criteria:

- Timing is not adversely affected
- Routing resources are available
- No overflow is created
- No design rule or process antenna violations are created

Widening wires helps avoid design rule violations, signal integrity problems, and yield loss.
Default: `1`

Note: To enable this parameter, you must also include a wide wire nondefault rule in the LEF file and specify the `-droutePostRouteWidenWireRule true` parameter

`-drouteMinSlackForWireOptimization` *`time_in_nanoseconds`*

Specifies the minimum slack on pins associated with nets that are candidates for wire spreading or wire widening. Using this parameter reduces the timing impact of wire spreading, wire widening, and via swapping, and helps reduce congestion and the yield loss due to shorts and opens.
Default: `0`

Note: To enable this parameter, you must also specify `-routeWithTimingDriven true`.

You can use the `-drouteMinSlackForWireOptimzation` parameter to reduce the timing closure effort during post-route ECO. NanoRoute will replace via to multi-cut via on non-critical nets, if a violation does not occur.

`-drouteNoTaperInLayers "`*`bottomLayerNum`*`:`*`topLayerNum`*`"`

Prohibits the detailed router from tapering in the specified layer range, creating non-default-width wires in that range.
Default: `""` (empty string)

`-drouteNoTaperOnOutputPin {true | false}`

Prohibits the detailed router from tapering at standard cells, macro cells, and block output pins.
Default: `false`

`-drouteOnGridOnly {none | via | all}`

Determines whether off-grid (off-track) routing is allowed.
Default: `none`

Note: For backward compatibility, this parameter also accepts `true` and `false` as arguments.

Specify one of the following arguments:

`all`	Does not allow off-grid routing of wires and vias.
`none`	Does not allow wires to be routed off grid.
`via`	Centers vias on routing grids by dropping them at track intersections.

`-droutePostRouteLithoRepair {true false}`

Corrects bridging, necking, and line-end shortening problems caused by lithography. Takes the severity of the problems into account and fixes the high-severity problems first. Prior to running the router with this parameter, import a hotspot interface format (HIF) file with `readHif` or `loadViolationReport`.
Default: `false`

Note: The design must be DRC clean before repairing lithography problems.

`-droutePostRouteMinimizeViaCount {true | false}`

Minimizes the number of vias on a fully routed design.

You do not have to specify a value for this parameter when you run the following command, because the software does so automatically:

`routeDesign -viaOpt`

Default: `false`

Note: Do not specify `true` for this parameter if you specify `true` for either of the following parameters:

- `-drouteUseMultiCutViaEffort`
- `-routeSelectedNetOnly`

`-droutePostRouteSpreadWire {true | false}`

Turns postroute wire spreading on (`true`) or off (`false`). Specify the minimum length for wires to move with `-drouteMinLengthForWireSpreading`. Use this parameter with the `routeDesign` command. For information, see "routeDesign" on page 1262.

The router moves the wires without moving the vias. Specifying `true` for this parameter might increase the overall routing run time but helps avoid yield lost caused by shorts.
Default: `false`

Note: This parameter is not enabled concurrently with postroute via swapping.

`-droutePostRouteSwapVia {multiCut | singleCut | none}`

Swaps single-cut vias for multiple-cut vias or reverses swapping on critical nets in a fully routed design, so that multiple-cut vias are swapped for single-cut vias on those nets. Does not swap multiple-cut vias in nondefault rule routing. In timing-driven routing, does not swap vias if doing so would negatively impact timing.

A net with a pin on a critical path is considered a critical net if the pin slack, plus the margin defined by the `-drouteMinSlackForWireOptimization` parameter, is less than or equal to 0.

Use this parameter with the `routeDesign` command. You do not have to specify a value for this parameter when you run the following command, because the software does so automatically:

`routeDesign -viaOpt`

For information, see "routeDesign" on page 1262.

Default: `none`

Important

This parameter is not enabled concurrently with postroute wire widening or wire spreading.

In addition, the software does not perform via swapping if `-droutePostRouteLithoRepair true` is specified.

Specify one of the following:

`multiCut`

Replaces single-cut vias with multiple-cut vias in a fully routed design.

`none`

Does not swap vias.

`singleCut`

During timing-driven routing, swaps vias on critical nets only; otherwise, might swap any multiple-cut via for a single-cut via. Swapping multiple-cut vias for single-cut vias can improve timing and relieve congestion as a postroute process.

`-droutePostRouteWidenWire {widen | shrink | none}`

Applies the rule specified by the `-droutePostRouteWidenWireRule` parameter to widen or shrink wires. Use this parameter with the `routeDesign` command. For information, see "routeDesign" on page 1262.
Default: `none`

Note: This parameter is not enabled concurrently with postroute via swapping.

Specify one of the following values:

`widen` — During timing-driven routing, widens only non-critical nets; otherwise widens as many nets as possible that are covered by the nondefault rule for widening wires.

`shrink` — During timing-driven routing, shrinks only critical nets; otherwise shrinks as many nets as possible that are covered by the nondefault rule for widening wires.

`none` — Turns off this parameter, so the router does not widen or shrink wires.

`-droutePostRouteWidenWireRule` *`ruleName`*

Specifies the nondefault rule for widening wires. The rule is specified in the LEF file. Use this parameter with the `routeDesign` command. For information, see "routeDesign" on page 1262.
Default: `""` (empty string)

`-drouteSearchAndRepair {true | false}`

Runs a search-and-repair step after the initial detailed routing (end iteration 20).
Default: `true`

Tip

Specify `false` for this parameter and run detailed routing to determine quickly if the design is routable.

`-drouteStartIteration` *`passNumber`*

Specifies the first pass in a detailed routing step. Together with `-drouteEndIteration`, divides detailed routing into intermediate steps.
Default: 0 (dynamic default setting)

Important

If the router has globally routed the design, but has not made any detailed routing passes, you must set `-drouteStartIteration` to `0` to build the initial detailed routing database.

`-drouteUseBiggerOverhangViaFirst {true | false}`

Specifies that the router choose vias with bigger metal overhang first during via optimization. the router chooses from fat single-cut and fat multiple-cut vias. All vias must be predefined or automatically generated before via optimization.
Default: `false`

`-drouteUseMultiCutViaEffort {low | medium | high}`

Specifies the effort level toward increasing the ratio of double-cut vias to single-cut vias concurrently with routing. Increasing the effort level increases the double-cut via ratio and decreases the total number of vias in the design, which means that it also decreases the number of single-cut vias. The lower the number of single-cut vias, the better the yield will be. After routing with this parameter specified, you can add more double-cut vias or larger overhang vias by using the `routeDesign` command. For examples, see routeDesign.

Default: `low`

Specify one of the following:

`low`	Specifies normal routing, so the router uses single-cut vias only.
`medium`	Balances the need for a high double-cut via ratio with run time and congestion. Specifying this parameter increases the run time somewhat compared with the `low` parameter. The router inserts some double-cut vias, although not as many as if the `high` parameter were specified.
`high`	Specifies the highest yield possible for vias, as the router puts its best effort toward achieving the highest possible double-cut via ratio at the expense of run time and congestion.

`-envAdvancedIntegration {true | false}`

Reduces the router's memory footprint in most designs, allowing it to route larger designs. This parameter is most effective when you are routing an unrouted design. Using this parameter when you route a design with a 32-bit machine allows the machine to handle large designs that could not be routed otherwise.
Default: `false`

`nano`	Nano Encounter™
`nanor`	NanoRoute™ Ultra
`nedbs`	SoC Encounter™ L (formerly called Nano Encounter DBS)
`soce`	SOC Encounter
`socegps`	SOC Encounter XL (formerly called SoC Encounter GPS)
`socegxl`	SoC Encounter GXL

`-envDontUseLicsForThreadings {"[nano][nanor][nedbs][soce] [socegps][socegxl]"}`

Important

This parameter is included for backward compatibility only. Cadence recommends that you use the `getMultiCpuLicense` command to specify the licenses to use for multi-CPU processing.

Excludes licenses from use for Superthreading or multi-threading. Enclose the arguments in double quotation marks if you specify more than one type of license. Set this parameter before the first global or detailed route command.
Default: `""` (empty string)

Specify one or more of the following options:

`nano`	Nano Encounter™
`nanor`	NanoRoute™ Ultra
`nedbs`	SoC Encounter™ L (formerly called Nano Encounter DBS)
`soce`	SOC Encounter
`socegps`	SOC Encounter XL (SoC Encounter GPS)
`socegxl`	SoC Encounter GXL

`-envNumberFailLimit` *`integer`*

Specifies the maximum number of error messages for any one command written to the log file before stopping processing.
Default: `1`

`-envNumberProcessor` *`numberProcessors`*

This parameter is included for backward compatibility. Cadence recommends that you use the `setMultiCpuUsage` command to specify the number of threads in muliple-CPU processing.

Specifies the maximum number of threads to use in multi-threading. Generally, using more threads decreases the run time. Specify this parameter before running any routing commands.

There is no absolute limit to the number of threads the router can use, however, Cadence recommends that you do not specify more threads than the number of CPUs available.

The router does not necessarily use the number of threads you specify—it determines the optimal number up to the value you specify. A small design generally uses fewer threads than a large design.
Default: `1`

Note: To enable a value greater than `1` for this parameter, you must have licenses available for the additional threads.

`-envNumberWarningLimit` *`integer`*

Specifies the maximum number of warning messages for any one command reported to the log file. After reaching this number, the router continues processing, but does not report warnings for this command.
Default: `20`

```
-envSuperthreading
   {"RSH {machineName number_CPUs nanoroute_executable} …" |
    "LSF {number_CPUs nanoroute_executable [bsub_options]} …" |
    "SGE {number_CPUs nanoroute_executable [qsub_options]} …" |
    "SSH {machineName number_CPUs nanoroute_executable} …" |
    "<option_configuration_file"}
```

Important

This parameter is included for backward compatibility. Cadence recommends that you configure Superthreading by using the `setDistributeHost` and `setMultiCpuUsage` commands.

Distributes detailed routing among several machines that can each have several processors. This parameter accepts a configuration file or an option string containing RSH, LSF, SSH, or SGE as arguments. Use an option string if you plan to use variables that you want the Tcl interpreter to parse. Use a configuration file if your option string contains Tcl meta characters and you do not want the Tcl interpreter to parse it. Specify this parameter before running any routing commands.
Default: To reset the NanoRoute router to run in non-Superthreading mode, enter the following command:
`setNanoRouteMode -envSuperthreading default`

Default: `""` (empty string)

Specify the following parameters:

`bsub_options`

Specifies the `bsub` options. For information on `bsub` options, see the LSF documentation.

`LSF` Specifies Load Sharing Facility. Client logs are sent by LSF through email. If you do not want to see the logs, specify the `-o` option. For more information on LSF, see the LSF documentation.

`machineName` Specifies a client machine.

`nanoroute_executable`

Specifies the path to the NanoRoute executable file.

number_CPUs Specifies the number of CPUs to use in the specified *machineName*.

option_configuration_file

Specifies a configuration file containing the LSF, RSH, SGE, or SSH parameters.

qsub_options

Specifies the qsub options. For information on qsub options, see the SGE documentation.

RSH Specifies the rsh (remote shell) command. You must be able to use remote shell from server to client machines without a password prompt. The NanoRoute router creates a log directory for debugging. The name of the log file is *machine*.log. For more information on rsh, see the rsh documentation.

SGE Specifies the Sun Grid Engine. For more information on SGE, see the SGE documentation.

SSH Specifies the Secure Shell. For more information on SSH, see the SSH documentation.

-help Outputs a brief description that includes type and default information for each setNanoRouteMode parameter.

For a detailed description of the command and all of its parameters, use the man command:
man setNanoRouteMode.

-reset Resets parameters to their default values. The -reset parameter *must* be the first parameter specified. If you specify -reset by itself, the software resets all setNanoRouteMode parameters to their default values. If you specify parameters after -reset, the software resets only those parameters to their default values.

`-routeAllowPowerGroundPin {true | false}`

Specifies targets for tie-high and tie-low nets.

- When `false`, connects tie nets to nearby straps, ring pins, or core rings.
- When `true`, connects tie nets to the same targets as when `false`, plus macro (power and ground) pins, cover macro pins, and standard cell followpins. I/O pad pins and pad ring pins are excluded from tie-net connections.

Default: `false` if the design contains straps. `true` if the design does not contain straps.

`-routeAntennaCellName {"`*`cellName`*`" | {"`*`list_of_cellNames`*`"}`

Specifies antenna diode cells to use during postroute optimization. The cells must have the same LEF `SITE` definition as the standard cells. Use this parameter with `-routeInsertAntennaDiode`.
Default: `""` (empty string) If you do not specify a cell, the router uses a core macro of type `ANTENNACELL` specified in the LEF `MACRO` statement.

`-routeAntennaPinLimit` *`numberPins`*

Specifies the maximum number of pins to consider on any net during process antenna violation repair.
Default: `1000`

`-routeAutoGgrid {true | false}`

Overwrites the `GCELLGRID` defined in the DEF file with a grid that is optimized for the NanoRoute router. Using this parameter helps resolve congestion and over-the-macro violations if you see routability issues due to hot spots.
Default: `true`

Note: This parameter is automatically set to `true` if `-routeWithEco` is `true`.

`-routeBottomRoutingLayer` *`layerNumber`*

Specifies the lowest layer the NanoRoute router uses for routing.

If the design has pins below the layer specified by this parameter, the router uses stacked vias to access those pins. However, if the tracks are not aligned, the router automatically creates off-grid tracks.
Default: `0`

Note: Use the `-bottom_preferred_routing_layer` attribute to set a soft limit for routing specific nets. For information, see "setAttribute" on page 1271.

`-routeConcurrentMinimizeViaCountEffort {low | medium | high}`

Specifies the effort level for reducing the total number of vias during routing. Use this parameter together with the `-drouteUseMultiCutViaEffort` parameter to control the effort toward reducing the total number of vias and the effort toward increasing the ratio of double-cut vias to single-cut vias concurrently with routing.

If you specify `-drouteUseMultiCutViaEffort medium` or `high`, the software sets the value of `-routeConcurrentMinimizeViaCountEffort` to `medium` unless you specify `-routeConcurrentMinimizeViaCountEffort yourself`. To realize further savings, set `-routeConcurrentMinimizeViaCountEffort` to `high`. This could further reduce the total via count up to 10 percent.

Note: Results might vary with your design.

Default: `low`

Specify one of the following parameters:

`low`	This is the default setting.
`medium`	Reduces total via count by 15 percent, with negligible runtime overhead.
`high`	Reduces the total via count by 20 percent to 25 percent, with marginal runtime overhead.

`-routeDeferredShield {true | false}`

Creates shared shield wires at the end of routing.

Before using this parameter, you must add a shield attribute to nets that need shielding. After routing, use the `createShield` command to add the shield wires. For more information, see "createShield" on page 1227.

Default: `false`

`-routeDeleteAntennaReroute {true | false}`

Deletes and reroutes nets with process antenna violations. This parameter is not enabled if you are using LEF 5.3 or older, or if the design has more than 100 DRC violations.
Default: `true`

Note: This parameter might cause a run-time penalty for congested designs.

`-routeDesignFixClockNets {true | false}`

Specifies whether to maintain the current status of clock nets when the `routeDesign` command runs or change the status to `ROUTED`.

- If you specify `true` for this parameter, `routeDesign` does not change the status of clock nets.
- If you do not specify this parameter (or you specify `false` for this parameter), `routeDesign` changes the status of clock nets changes to `ROUTED`.

Default: `false`

The parameter applies to the `routeDesign` command only—not to `globalRoute`, `detailRoute`, or `globalDetailRoute`.

`-routeDesignNoCheckPlace {true | false}`

Specifies whether the `routeDesign` command skips `checkPlace` before running.
Default: `false`

`-routeDesignRouteClockNetsFirst {true | false}`

Detects whether clock nets require ECO routing after an `optDesign -postCTS` run in which some of the registers might have been moved before routing other nets during an ECO flow. When you specify `true` for this parameter, the `routeDesign` supercommand checks whether the existing clock nets were fully or partially routed with the NanoRoute router and reroutes them before routing the remaining nets. If the existing clock nets were not routed with NanoRoute, `routeDesign` skips routing on those nets.
Default: `false`

`-routeEcoOnlyInLayers "`*`bottomLayerNum`*`:`*`topLayerNum`*`"`

Specifies the bottom and top layers for ECO routing. Routing on all other layers is frozen. For example, `-routeEcoOnlyInLayers "3:5"` means that the NanoRoute router routes only on layers 3, 4, and 5. In this example, a `via23` can be used to access a *metal2* pin if the via is completely enclosed in the *metal2* pin layer.
Default: `""` (empty string)

`-routeExtraViaEnclosure` *`distance`*

Specifies an extra via enclosure to use when connecting to block pins and special wires. To center vias for via access, specify a value that is equal to one-half the width of the largest pin or wire. An extra enclosure of this size forces NanoRoute to connect tie-high and tie-low nets to the center of power routes.
Default: `0` (microns)

`-routeFixTopLayerAntenna {true | false}`

Reserves the top routing layer for repairing process antenna violations.

When this parameter is `true`, the NanoRoute router globally routes nets that do not have diode protection, such as nets without output pin protection, without using the top routing layer. During detailed routing, it repairs antenna violations by jumping to the highest layer. This is useful for routing feedthrough nets.
Default: `true`

`-routeHonorPartition {true | false}`

Routes nets belonging to a partition completely inside that partition and nets that are part of the top-level glue logic completely within the top-level channels.
Default: `false`

`-routeHonorPowerDomain {true | false}`

Routes nets without crossing power domains.

- If a net connects pins that all belong to one power domain, the software tries to route the net within that power domain. If completing the routing is not possible without breaking the rules, due to congestion for example, completes the routing and issues a warning message.
- If a net connects pins that belong to different power domains, the software tries to route the net within the involved power domains, without crossing over into other power domains that do not include any of the pins in the net. If completing the routing is not possible without breaking this rule, completes the routing and issues a warning message.

Default: `false`

You must specify a power domain before using this parameter. For information on specifying power domains, see the following documents:

- *Common Power Format Language Reference*
- *Common Power Format User Guide*

`-routeIgnoreAntennaTopCellPin {true | false}`

Ignores antenna violations on top-level I/O block pins, but repairs antenna violations elsewhere. Specify `false` for this parameter when you route a block whose top level is going to have diodes inserted.
Default: `true`

`-routeInsertAntennaDiode {true | false}`

Inserts and places antenna diode cells if there are available placement locations. By default, the NanoRoute router repairs antenna violations by changing layers (also called antenna stapling or layer hopping), but it can also repair antenna violations by inserting diode cells as close as possible to input gates to discharge current.

The router inserts antenna diode cells even if filler cells are already placed. It can swap filler cells with antenna diode cells and fill the gap automatically if an antenna diode cell is not the same size as the filler cell it replaced.
Default: `false`

- To specify the diode cells, use `-routeAntennaCellName`.

  You can specify only one cell at a time. If you do not specify a cell, the router uses a core macro of type `ANTENNACELL` specified in the LEF `MACRO` statement.

`-routeInsertAntennaInVerticalRow {true |false}`

Allows the router to place antenna diodes in vertical rows.
Default: `false`

`-routeInsertDiodeForClockNets {true | false}`

Inserts diodes to repair process antenna violations on clock nets that are in the regular net section of the DEF file.
Default: `false`

`-routeMergeSpecialWire {true | false}`

Merges overlapping same-net special wires to create large rectangles when calculating spacing requirements. Merging might be required if the result of merging produces shapes that fall on a different design-rule width range than the original shapes.
Default: `false`

Note: Use of this parameter is recommended only for designs that require special wire merging. Using this parameter requires additional CPU time. Contact your Cadence representative for more information.

`-routeMinShieldViaSpan` *`float`*

Specifies the span distance (in microns) between vias on shielding nets (VSS or VDD) while connecting to the power structure. This parameter sets a soft limit.
Default: `-1` (infinite. NanoRoute will drop one via.)

`-routeReverseDirection "(`*`x1 y1 x2 y2`*`) (`*`x3 y3 x4 y4`*`) ...."`

Reverses the preferred routing direction in the specified areas. You can specify one or more areas.

> **Note:** To route a specified net in the nonpreferred direction in the specified area, you must also use the following commands:
>
> ```
> selectNet netName
> setNanoRouteMode -routeSelectedNetOnly true
> ```

You can also use this parameter to increase the utilization of routing channels where some routing layers are blocked by power routing. For example, in a narrow routing channel, the long way usually has more traffic than the crosswise direction. If the preferred direction for power routing is the long way, the power routing might use too many routing resources and cause congestion. Reversing the preferred routing direction in this channel, that is, using routing layers without power wires for long way routing, might increase available routing resources.

`-routeSelectedNetOnly {true | false}`

Specifies whether NanoRoute routes all nets at once or routes selected nets only. To route critical nets as short as possible, select the critical nets and set this parameter to `true`. NanoRoute does the following:

- Removes incomplete nets unless they are marked `+ FIXED` or have a `-skip_routing` net attribute.
- Routes the remaining selected nets.

Default: `false`

Tip

To select a net, use the `selectNet` command or click the net in the design display window. For information on `selectNet`, see "selectNet" on page 556.

`-routeSiEffort {high | medium | low | min | normal | max}`

Specifies how aggressively the router works to minimize crosstalk. Works in conjunction with `-routeWithTimingDriven`.
Default: `""` (dynamic default setting)

- When timing-driven routing is specified, the default value is `high`. You can also specify `medium` or `low`. If you specify `min`, `normal`, or `max`, the software converts it to `high` and issues a warning message.

  A `high` signal integrity effort maximizes the priority for preventing and reducing crosstalk. It requires more time during detailed routing and consumes more routing resources than a `medium` or `low` effort.

- When timing-driven routing is not specified, the default value is `min`. You can also specify `normal` or `max`. If you specify `low`, `medium`, or `high`, the software converts it to the appropriate value (`low` is converted to `min`; `normal` is converted to `medium`; `high` is converted to `max`).

  The `min`, `normal`, and `max` parameters are included for backward compatibility with previous versions of the software.

Note: You must specify `-routeWithSiDriven` to enable `-routeSiEffort`.

`-routeStrictlyHonorNonDefaultRule {true | false}`

Uses nondefault spacing rules for nondefault wires.

> **Note:** The router tapers wires and uses default vias even when this parameter is set to `true`.

By default, the router treats nondefault rule spacing as a soft attribute; that is, when routing resources are available, it honors nondefault rules as they apply to wiring width, design rule spacing and via size. If an area is too congested, and resources are not available, the router might not honor nondefault spacing. When you specify `true` for this parameter the router honors nondefault spacing but still does not change its behavior with regard to nondefault wiring width and via size.

You define nondefault routing rules in the `NONDEFAULTRULE` statement of the LEF file.

Default: `false`

`-routeStripeLayerRange "`*`bottomLayerNum`*`:`*`topLayerNum`*`"`

Specifies the range of routing layers for stripes.
Default: `""` (empty string)

`-routeTdrEffort` *`integer`*

Specifies how aggressively the router works to meet timing constraints. A higher value increases the effort toward meeting timing constraints and decreases the effort toward relieving congestion.
Default: 2 (Dynamic default setting: `globalDetailRouteBatch` defaults to `0`; `globalRoute`, `detailRoute`, and `globalDetailRoute` are tuned automatically by the software.)
Range: `0-10`

Note: You must specify `-routeWithTimingDriven` to enable `-routeTdrEffort`.

`-routeTopRoutingLayer` *`layerNumber`*

Specifies the highest layer the router uses for routing. Use this parameter to limit the routing layers for blocks or to reserve the top layers for power and ground stripes. This parameter sets a hard limit; that is, the router does not route nets above the specified layer. To set a soft limit, specify the `-top_preferred_routing_layer` attribute for specific nets. For information on `-top_preferred_routing_layer`, see “setAttribute” on page 1271.

Specifying a maximum routing layer with this parameter also changes the maximum routing layer setting in the `setTrialRouteMode` command. For example, if you specify:

```
setNanoRouteMode -routeTopRoutingLayer 5
```

it also specifies the following `setTrialRouteMode` setting:

```
setTrialRouteMode -maxRouteLayer 5
```

Likewise, if you specify a maximum routing layer using the `setTrialRouteMode` command, that value is applied to the `setNanoRouteMode` command.
Default: 0 (Layer specified by the `setMaxRouteLayer` command)
Range: `0-15`

Note: The layer you set with this parameter must be equal to or lower than the layer specified by the `setMaxRouteLayer` command. For more information, see “setMaxRouteLayer” on page 1279.

`-routeTrunkWithClusterTargetSize` *`integer`*

Controls the routing pattern of nets with the `trunk` pattern attribute. By default, the router connects each pin in nets on which you have applied `setAttribute -pattern trunk` directly to the trunk of the special net; use this parameter to change the routing pattern for those nets:

- Specify 0 to route nets with the `trunk` pattern specified with a Steiner tree pattern instead of a trunk pattern.
- Specify an integer greater than 1 to route the nets with a fishbone pattern. In a fishbone pattern, several pins in a column are clustered together before connecting to the special net. The value of this parameter represents the approximate number of pins in each cluster.

Default: `1`

`-routeUseBlockageForAutoGgrid {true | false}`

Generates global routing cells (gcells) aligned to blockages in macros. Use this parameter prior to global routing, to improve the alignment of gcells with pins.
Default: `false`

`-routeWithEco {true | false}`

Routes in engineering change order (ECO) mode if `globalDetailRoute` or `globalDetailRouteBatch` is specified. During ECO mode, NanoRoute completes partial routes with added logic, while maintaining the existing wire segments as much as possible. When `-routeWithEco` is `true`, the router sets the `-routeAutoGgrid` parameter to `true`.
Default: `false`

Note: The Encounter software also has an `ecoRoute` command. For more information, see "ecoRoute" on page 254.

`-routeWithLithoDriven {true | false}`

Avoids lithography problems during routing by avoiding certain routing patterns that might lead to the creation of lithography hotspots.
Default: `false`

`-routeWithSiDriven {true | false}`

Prevents or reduces crosstalk. Works in conjunction with timing-driven routing.

When timing-driven routing is specified, uses SMART routing to identify victim nets and minimize crosstalk by wire spacing, layer hopping, net ordering, and minimizing the use of long parallel wires.

When timing-driven routing is not specified, uses an older signal integrity engine to minimize crosstalk by preventing or reducing the use of long parallel wires.
Default: `false`

- Specify the crosstalk prevention effort with `-routeSiEffort`.
- Specify `-routeWithTimingDriven` and `-routeWithSiDriven` to run SMART routing.

`-routeWithSiPostRouteFix {true | false}`

After routing and signal integrity analysis, uses information from a file (that contains a list of victim nets) to change routing topology, including layer assignments, spacing, and location, to avoid coupling to other nets. This parameter applies to nets with the `-si_post_route_fix` attribute set to `true`.
Default: `false`

Note: Postroute signal integrity repair requires that you run both global and detailed routing.

Note: The NanoRoute router can use a file from a crosstalk analysis tool, such as the CeltIC™ crosstalk analyzer for cell-based designs. For information on using CeltIC commands in the Encounter environment, see “Signal Integrity Commands” on page 1511.

`-routeWithTimingDriven {true | false}`

Minimizes timing violations by analyzing the timing slack for each path, the drive strengths of each cell in the library, and the maximum capacitance and maximum transition limits. During timing-driven routing, NanoRoute routes multi-pin nets to the most critical sink first, performs wire optimization by reducing resistance and coupling, and continually adjusts detouring.

When this parameter is set to `true`, the router automatically generates a timing information file named `.timingfile.tif` in the working directory. It uses this file when it runs timing driven routing.

Note: Specify the following commands and to skip the CTE call and timing file generation step and use the specified file:

```
writeDesignTiming fileName
setNanorouteMode -routeWithTimingDriven true
setNanoRouteMode -timingEngine fileName
```

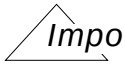
Default: `false`

- Specify the timing-driven effort with `-routeTdrEffort`.
- Specify the timing engine or a timing graph with `-timingEngine`.
- Specify `-routeWithTimingDriven` and `-routeWithSiDriven` to run SMART routing.

You can use the `-routeWithTimingDriven` parameter to reduce the timing closure effort during post-route ECO. NanoRoute will replace via to multi-cut via on non-critical nets, if a violation does not occur.

`-routeWithViaInPin {true | false | bottomLayerNum:topLayerNum}`

Encloses via geometries completely inside standard cell pins. Specify `true` for this parameter to avoid `MINSTEP` violations for standard cell pin access.
Default: `false`



To avoid violations, specify the same value for this parameter throughout the entire design flow.

`true` Enclose vias inside standard cell pins on all layers.

`false` The router is not required to enclose vias inside standard cell pins.

`bottomLayerNum``:`*`topLayerNum`*

Enclose vias inside standard cell pins on layers in the specified layer range.

- `1:1`

  Specifies that *M1* pins must enclose vias.

- `2:5`

  Specifies that pins on *M1*, *M2*, and *M3* must enclose vias. Pins on other layers are not required to enclose vias.

`-routeWithViaOnlyForStandardCellPin {true | false |` *`bottomLayerNum`*`:`*`topLayerNum`*`}`

Allows standard cell pins to be accessed by vias only. Prohibits planar access.
Default: `false`

`true` Always use vias for standard cell pin access on all layers.

`false` Use vias or planar access for standard cell pins on all layers.

`bottomLayerNum``:`*`topLayerNum`*

Always use vias for pin access in the specified layer range.

- `1:1`

  Always use vias for access to *metal1* standard cell pins. *metal2* and above can use vias or planar access.

- `2:5`

  Always use vias for access to standard cell pins on *metal2*, *metal3*, *metal4*, and *metal5*. *metal1* and *metal6* (and above) can use vias or planar access.

Important

Set this option before any routing or verification because it affects the pin access preparation steps. In addition, after you set this option, the access remains for same the entire Encounter session. Do not change the setting, even for future routing sessions that load saved data from the database or a saved DEF file.

For example, if a design with pins on *metal1* and *metal2* is routed with this option set to `1:1`, the router uses vias for access to all standard cell pins on *metal1*, and vias or planar access to standard cell pins on *metal2*. If you save the design after routing, then import the design in a different Encounter session and route it again with this option set to `true` (that is, forcing the router to use vias to access all standard cell pins on all layers), you might have planar connections (from the previous routing session) to pins on *metal2*. All the planar connections to those pins become violations, because they are not allowed when this option is set to `true`.

`-timingEngine {CTE | externalTimingGraph}`

Specifies the timing engine NanoRoute uses for timing analysis.
Default: `CTE`

Specify one of the following:

`CTE` Specifies the Encounter common timing engine.

externalTimingGraph

Specifies an externally generated timing graph.

For more information on timing-driven routing with the NanoRoute router, see "Running Timing-Driven Routing" in the *Encounter User Guide*.

Examples

- The following commands show the steps in basic routing strategy with the NanoRoute router.

  a. The router globally routes the design:

  ```
  globalRoute
  ```

  b. The router does the initial detailed routing (iteration 0 does not include a search-and-repair step), and saves the design as `droute0`:

  ```
  setNanoRouteMode -drouteStartIteration 0
  setNanoRouteMode -drouteEndIteration 0
  detailRoute
  saveDesign droute0
  ```

  c. The router does the first search-and-repair iteration, and saves the design for analysis:

  ```
  setNanoRouteMode -drouteStartIteration 1
  setNanoRouteMode -drouteEndIteration 1
  detailRoute
  saveDesign droute1
  ```

  d. The router does the second to nineteenth search-and-repair iterations, and saves the design for analysis. The switch box grows larger as the iteration number increases.

  ```
  setNanoRouteMode -drouteStartIteration 2
  setNanoRouteMode -drouteEndIteration 19
  detailRoute
  saveDesign droute19
  ```

  e. The router runs postroute optimization (`drouteEndIteration default`) and additional search-and-repair operations, and saves the design as `droute`:

  ```
  setNanoRouteMode -drouteStartIteration 20
  setNanoRouteMode -drouteEndIteration default
  detailRoute
  saveDesign droute
  ```

- The following commands route selected nets, then route the remaining nets in the design:

```
setNanoRouteMode -routeSelectedNetOnly true
globalDetailRoute
setNanoRouteMode -routeSelectedNetOnly false
globalDetailRoute
```

- The following commands shows the basic strategy for repairing process antenna violations using the NanoRoute router:

  ```
  setNanoRouteMode -drouteFixAntenna true
  setNanoRouteMode -routeAntennaDiodeCellName "ANTENNA"
  setNanoRouteMode -routeInsertAntennaDiode true
  globalDetailRoute
  ```

  The router runs global and detailed routing. It repairs as many process antenna violations as it can by layer hopping during the search-and-repair step and postroute optimization. If any process antenna violations remain, the router repairs them by inserting antenna diode cells named `ANTENNA`.

- The following commands repair process antenna violations by inserting diodes and insert filler cells from the `my_fillers.txt` file to fill in the gaps left when the diodes replace large filler cells:

  ```
  setNanoRouteMode -routeInsertAntennaDiode true
  setNanoRouteMode -routeAntennaCellName "cell_A"
  setNanoRouteMode -routeReInsertFillerCellList my_fillers.txt
  globalDetailRoute
  ```

- The following commands route a placed design in timing-driven mode and output a routed DEF file:

  ```
  restoreDesign
  setNanoRouteMode -routeWithTimingDriven true
  saveDesign init
  globalDetailRoute
  saveDesign droute
  defOut -floorplan -placement -routing droute.def
  ```

- The following command resets the `-routeSelectedNet` parameter to its default value:

  ```
  setNanoRouteMode -reset -routeSelectedNet
  ```

- The following command resets all `setNanoRouteMode` parameters to their default values:

  ```
  setNanoRouteMode -reset
  ```

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*

setTrialRouteMode

```
setTrialRouteMode
     [-help]
     [-reset]
     [-autoSkipTracks {true | false}]
     [-blockageCostMultiple integer]
     [-detour {true | false}]
     [-excludePartition {fileName}]
     [-extendM1PinToM2 {true | false}]
     [-fastRouteForPinAssign {true | false}]
     [-floorPlanMode {true | false}]
     [-handleEachPatition {true | false}]
     [-handleEachPD {true | false}]
     [-handlePartFixedNets {true | false}]
     [-handlePartition {true | false}]
     [-handlePartitionComplex {true | false}]
     [-handlePD {true | false}]
     [-handlePDComplex {true | false}]
     [-handlePreroute {true | false}]
     [-highEffort {true | false}]
     [-honorNetRTLayer {true | false}]
     [-honorPin {true | false}]
     [-honorRoutingHalo {true | false}]
     [-honorActiveLogicView {true | false}]
     [-ignoreAbutted2TermNet {true | false}]
     [-ignoreBumpNets {true | false}]
     [-ignoreGuideLayer {true | false}]
     [-ignoreNetIsSpecial {true | false}]
     [-ignoreObstruct {true | false}]
     [-ignorePrefExtraSpace {true | false}]
     [-ignoreRTBlockage {fileName}]
     [-intraNets {true | false}]
     [-keepEndPoints {true | false}]
     [-keepEndPoints2 {true | false}]
     [-keepExistingRoutes {true | false}]
     [-keepPreferredLayer {true | false}]
     [-maxDetourRatio float]
     [-maxRouteLayer {layerIndex | layerName}]
     [-minRouteLayer {layerIndex | layerName}]
     [-PDAwareSelNet {true | false]]
     [-pinGuide {true | false}]
     [-PKS {true | false}]
     [-printSections {true | false}]
     [-printWiresOnRTBlk {true | false}]
     [-printWiresOnRTBlkFile {fileName}]
     [-printWiresOnRTBlkLong {true | false}]
     [-printWiresOnRTBlkLongFile {filename}]
     [-printWiresOutsideBusguide {true | false}]
     [-ptnBdryExt float]
     [-ptnBdryShr float]
```

```
[-routeGuide {fileName}]
[-routeObs {true | false}]
[-selMarkedNet {true | false}]
[-selMarkedNetOnly {true | false}]
[-selNet {fileName}]
[-selNetOnly {fileName}]
[-skipTracks {"layerName tracksToSkip:TotalTracks"}]
[-updateRemainTrks {true | false}]
[-useM1 {true | false}]
```

Sets global parameters for the Trial Route program. Parameters that you specify with `setTrialRouteMode` are then used automatically whenever you run Trial Route, either explicitly through the `trialRoute` command or Trial Route form, or internally during in-place optimization, partitioning, or placement. The `setTrialRouteMode` parameters affect the behavior of the following commands:

- `placeDesign`
- `optDesign`

Use the `getTrialRouteMode` command to return the current settings for the `setTrialRouteMode` command.

Parameters

`-autoSkipTracks {true | false}`

Classifies nets by nondefault rule, then route the nets for each rule, using the track constraints specified in the rule. If you specify `-autoSkipTracks true`, Trial Route routes nets from the rule with the largest pitch first, then the second largest pitch, and so forth. After routing the nets for a rule, Trial Route marks them as pre-routed before routing the nets of the next rule.

Note: Wire overlap can occur if a gcell is only partially covered by a routing blockage or pre-routed wires.

Default: `false`

`-blockageCostMultiple` *`integer`*

Multiplies the cost of the wire by the specified value in order to avoid overlapping with a routing blockage in an area that is not completely blocked. The higher the wire number, the more Trial Routes attempts to avoid overlapping the wire with a blockage.

Use this parameter when there are long, thin routing blockages. You can specify a value between `1` and `100`.

Default: `1`

`-detour {true | false}`

Does not route through congested areas.

Note: You should not use `-detour false` for a production design. The routing of the design is not accurate because some nets are routed through congested areas or over blockages in order to complete the routing. Specify `-detour false` only when routing a poorly designed netlist in which a part of the design is good and needs to be analyzed.

Default: `true`

`-excludePartitionFile {`*`fileName`*`}`

Routes around user-specified partitions when routing nets belonging to more than one partition.

When you specify `-excludePartitionFile true` with `-handlePartitionComplex true` (or with `-handlePartition true`), Trial Route only takes into account the partitions listed in the specified file, when routing nets belonging to more than one partition. Trial Route does not have to detour around partitions not specified in the file, and they can be used for feedthrough insertion.

Default: `" "` (empty string)

`-extendM1PinToM2 {true | false}`

Reserves one *metal2* track for use by a *metal1* pin in the same gcell. When Trial Route routes the *metal1* pin, it uses that *metal2* track to route to *metal2*.

Specify `-extendM1PinToM2 true` to achieve a more accurate model of *metal2* congestion.

Default: `false`

`-fastRouteForPinAssign {true | false}`

Creates a partition-aware routing topology for approximately 95 percent of inter-partition nets that is similar to the routing topology generated by the `-handlePartitionComplex` parameter.

Trial Route routes net groups that have a significant number of inter-partition nets in a partition-aware manner, using larger gcells. The remaining net groups with fewer inter-partition nets are routed in a manner similar to flat Trial Route. Using this method, Trial Route can generate the routing topology faster than with the `-handlePartitionComplex` parameter.

To also route intra-partition nets, specify `-intraNets true` in conjunction with this parameter.

Note: The `-fastRouteForPinAssign` parameter should be used only for performing congestion-aware pin assignment.

Default: `false`

`-floorPlanMode {true | false}`

Runs Trial Route more quickly by increasing the size of the global routing cells with fewer iterations.

Default: `false`

`-handleEachPartition {true | false}`

Routes each intra-partition net set one at a time in order to avoid intra-partition nets being routed through other partitions that are located inside.

When two partitions abut each other or are too close, a routing blockage cannot be generated. As a result, an intra-partition net can be routed through another partition located inside of the partition. Use `-handleEachPartition true` to block the net from crossing the boundary of the other partition.

Note: You must specify `-handlePartition true` or `-handlePartitionComplex true` in order to use this parameter.

Default: `false`

`-handleEachPD {true | false}`

Routes each intra-power domain net set one at a time in order to avoid intra-power domain nets being routed through other power domains that are located inside.

When two power domains abut each other or are too close, a routing blockage cannot be generated. As a result, an intra-power domain net can be routed through another power domain located inside of the power domain. Use `-handleEachPD true` to block the net from crossing the boundary of the other power domain.

Note: You must specify `-handlePD true` or `-handlePDComplex true` in order to use this parameter.

Default: `false`

`-handlePartFixedNets {true | false}`

Re-routes partially routed nets with `FIXED` or `COVER` wires.

When you specify `-handlePartFixedNets true`, if your DEF file contains partially-routed nets with `FIXED` or `COVER` wires, Trial Route completes the routing for the nets. Trial Route cannot maintain the exact location of the wires; however, they are accurate within the gcell area. Also, if you write out a DEF file after routing, all wires for the net are marked `FIXED`. You can no longer identify the original `FIXED` wires.

Trial Route does not re-route completely routed nets with `FIXED` or `COVER` wires.

Default: `false`

`-handlePartition {true | false}`

Routes intra-partition and global nets using the specifications of the partitions, such as area and the number of layers. This routing is based on the resources available after the chip is partitioned.

Specify `-handlePartition true` to gauge the congestion at the full chip level for the chip to be partitioned.

For flat partitioned designs, you can specify `-handlePartition true` to simulate channel-based routing. For channelless designs, you must perform feedthrough insertion before you can use `-handlePartition true`. For more information, see "Routing a Partitioned Design" in the *Encounter User Guide*.

Note: Specifying `-pinGuide true` affects the setting of the `-handlePartition` parameter. If `-handlePartition false` is set, and you specify `-pinGuide true`, Trial Route ignores the `-handlePartition false` setting, and runs as if `-handlePartition true` is specified instead. If you want to use `-handlePartition false`, you must specify `-pinGuide false`.

Default: `false`

`-handlePartitionComplex { true | false}`

Routes a net belonging to more than one partition so that the routing does not violate partition boundaries, in addition to routing intra-partition and global nets using the specifications of the partitions. For example, if a net belongs to partitions P1 and P3, but not P2, the net is routed without going over the P2 partition boundary.

For flat partitioned designs, you can specify `-handlePartitionComplex true` to simulate channel-based routing. For channelless designs, you must perform feedthrough insertion before you can use `-handlePartitionComplex true`. For more information, see "Routing a Partitioned Design" in the *Encounter User Guide*.

Default: `false`

`-handlePD {true | false}`

Routes intra-power domain and global nets using the specifications of the power domains. If you specify `-handlePD true`, Trial Route routes intra-power domain nets only within their power domains. Global nets that do not connect with a power domain do not get routed through that power domain.

Default: `false`

`-handlePDComplex {true | false}`

Routes nets belonging to more than one power domain so that the routing does not violate power domain boundaries, in addition to routing intra-power domain and global nets using the specifications of the power domains.

For example, when you specify `-handlePDComplex true`, if a net belongs to power domains `PD1` and `PD3`, but not `PD2`, Trial Route routes the net without going over the `PD2` power domain boundary.

Default: `false`

`-handlePreroute {true | false}`

Does not re-route nets with a database status of `IsIgnoreInRoute`, but re-routes all other nets.

You can use the `dbIsNetIgnoreInRoute` database command to check if a net has a database status of `IsIgnoreInRoute`. To set the status of a net to `IsIgnoreInRoute`, use the `dbSetIsNetIgnoreInRoute` database command.

Note: Trial Route automatically maintains completely routed nets marked as `FIXED` or `COVER`.

Default: `false`

`-help`

Outputs a brief description that includes type and default information for each `setTrialRouteMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setTrialRouteMode`.

`-highEffort {true | false}`

Runs additional iterations for resolving difficult routing congestions.

Default: `false`

`-honorActiveLogicView {true | false}`

Honors Active Logic View. When you specify `-honorActiveLogicView true`, instead of routing the whole chip, the software routes only the nets related to interface paths between the partitions masked by Active Logic View.

Note: Trial Route only supports this parameter before CTS is run.

Default: `false`

`-honorNetRTLayer {true | false}`

Honors the maximum and minimum routing layer limits specified for individual nets using one of the following commands:

```
setAttribute
    -bottom_preferred_routing_layer number
    -top_preferred_routing_layer number

setCTSMode
    -routeTopPreferredLayer number
    -routeBottomPreferredLayer number
```

Default: `true`

`-honorPin {true | false}`

Follows pre-assigned pins (pins marked `FIXED`) and assigned pins (pins marked `PLACED`). When you specify `-honorPin true`, if these pins exist, Trial Route automatically calls `-handlePartition true` to keep wires within or outside partition boundaries. If there are no existing pre-assigned or assigned pins, Trial Route runs once, which is its default behavior.

Default: `false`

`-honorRoutingHalo {true | false}`

Limits routing inside a routing halo in order to avoid cross coupling between routing inside and outside of the block. If you specify `-honorRoutingHalo true`, Trial Route considers it high cost to route inside a routing halo, and only routes inside the halo area to connect to pins on the boundary of the block.

Default: `true`

`-ignoreAbutted2TermNet {true | false}`

Does not route nets that have two pins on the same layer whose pin geometries abut (that is, touch each other).

Default: `false`

`-ignoreBumpNets {true | false}`

Does not route nets with connections to bumps. Nets with connections to bumps are typically routed using the `fcroute` command. Use `-ignoreBumpNets true` to prevent Trial Route from routing these nets.

Default: `false`

`-ignoreGuideLayer {true | false}`

Ignores the layer information in a route guide file, if one is specified, and determines the wire layers during routing. The wire layer information specified in the route guide file includes the coordinates of the wire ends, whether it is a horizontal or vertical wire, the width of the wire, and the layer number.

Default: `false`

`-ignoreNetIsSpecial {true | false}`

Does not route nets that have a status of Special in the database. Trial Route does not route special nets, but considers existing special wires when calculating congestion.

You can use the `dbIsNetSpecial` database command to check if a net has a database status of Special.

Default: `false`

`-ignoreObstruct {true | false}`

Routes over all blocks and partitions, instead of just those that do not have routing blockages specified in the LEF file.

Default: `false`

`-ignorePrefExtraSpace {true | false}`

Ignores the extra net spacing specified for individual nets with the `setAttribute` and `setCTSMode` commands.

Default: `false`

`-ignoreRTBlockage {`*`fileName`*`}`

Ignores routing blockages within a specified area to avoid creating detours around them. When you specify `-ignoreRTBlockage true`, Trial Route reads the specified file, and ignores all routing blockages and cell obstructs that are within the box areas defined in the file.

Use the following format to define the box areas in the file:

```
layername (llx,lly urx,ury)
```

For example:

```
M3 (288.431,279.117 377.079,648.289)
```

Default: `" "` (empty string)

`-intraNets {true | false}`

When specified with `-fastRouteForPinAssign true`, routes intra-partition nets, in addition to routing inter-partition and top-level nets.

Note: This parameter must be specified in conjunction with the `-fastRouteForPinAssign` parameter. The `-fastRouteForPinAssign` and `-intraNets` parameters should be used only for performing congestion-aware pin assignment.

Default: `false`

`-keepEndPoints {true | false}`

Preserves only the floating wires marked with an “F” in the guide file, and completes their routing. All other floating guide file wires are deleted.

For example, if you specify `-keepEndPoints true`, the software preserves the following floating guide file wire:

```
wire 3819.3 -289.5 3827.1 -289.5 H 0.5 L 2 F
```

Currently, you must manually add the “F” in the guide file to mark the floating wires that you want to preserve.

Note: You must specify a guide file with the `-guide` parameter in order to use `-keepEndPoints true`.

Default: `false`

`-keepEndPoints2 {true | false}`

Preserves all floating wires in the guide file and completes their routing.

Note: You must specify a guide file with the `-guide` parameter in order to use `-keepEndPoints2 true`.

Default: `false`

`-keepExistingRoutes {true | false}`

Preserves all existing partially and completely pre-routed nets.

Note: Trial Route automatically maintains partially and completely routed nets marked as `FIXED` or `COVER`.

Default: `false`

`-keepPreferredLayer {true | false}`

Attempts to route short pin connections from long wires by routing in the preferred layer direction on layers. For example:

M3(H) -> M2(V) -> M1(pin)

If you specify `-keepPreferredLayer false`, Trial Route can create the connections by routing in both the horizontal and vertical directions on a layer. For example:

M2(H) -> M2(V) -> M1(pin)

Default: `false`

`-maxDetourRatio` *`float`*

Attempts to re-route two-pin nets to reduce wire length when the wire length is greater than:

float x half the net bounding box perimeter

Trial Route reduces the wire length by relaxing the cost for congested gcells; therefore, reducing wire length can increase congestion.

Specify `-maxDetourRatio true` when there are two-pin nets with long detours that slow timing. This parameter applies only to two-pin nets.

You can specify a value between `0.0` and `100.0`.

Default: `0`

`-maxRouteLayer {`*`layerIndex | layerNumber`*`}`

Limits routing to layers at and below the specified layer. All metal layers above the specified metal number are not routed.

The value you specify must be less than or equal to any maximum routing layer limit set with the setMaxRouteLayer command. If you specify a value that is greater than the `setMaxRouteLayer` value, Trial Route automatically resets the value to be equal to the `setMaxRouteLayer` value.

Specifying a maximum routing layer with this parameter also changes the maximum routing layer setting in the setNanoRouteMode command. For example, if you specify:

```
setTrialRouteMode -maxRouteLayer 5
```

it also specifies the following setting for the NanoRoute router:

```
setNanoRouteMode -routeTopRoutingLayer 5
```

Likewise, if you specify a maximum routing layer using the `setNanoRouteMode` command, that value is applied to the `setTrialRouteMode` command.

Default: `15`

`-minRouteLayer {`*`layerIndex`* `|` *`layerNumber`*`}`

Limits routing to layers at and above the specified layer. However, layers below the specified layer can still be used for routing if necessary, such as for short connections to pins.

You must specify a number greater than 1. Specify `-useM1 true` to use layer *metal1* for routing.

Specifying a minimum routing layer with this parameter also changes the minimum routing layer setting in the setNanoRouteMode command. For example, if you specify:

```
setTrialRouteMode -minRouteLayer 3
```

it also specifies the following setting for the NanoRoute router:

```
setNanoRouteMode -routeBottomRoutingLayer 3
```

Likewise, if you specify a minimum routing layer using the `setNanoRouteMode` command, that value is applied to the `setTrialRouteMode` command.

Note: If you specify `-minRouteLayer` and `-selNet true`, Trial Route routes the nets in the file first on the layers at and above the specified minimum routing layer, then routes the remaining nets on all layers.

Default: `0`

`-PDAwareSelNet {true | false}`

When specified with `-handlePD true` and `-selNet` *`filename`*, routes only the intra-power domain nets that are specified in *`filename`* within their power domains. Global nets that do not connect with a power domain are then routed outside of the power domains.

Note: This parameter must be specified in conjunction with the `-handlePD` and `-selNet` parameters.

Default: `false`

`-pinGuide {true | false}`

Uses the pin guide statements in the floorplan file to guide the routing through partition pin points.

If you specify `-pinGuide true`, the software uses the pin guide statements in the floorplan file, if they exist. If pin guide statements exist, Trial Route automatically calls `-handlePartition true` to keep wires within or outside partition boundaries. If there are no existing pin guide statements, Trial Route runs once, which is its default behavior.

Note: Specifying `-pinGuide true` parameter affects the setting of `-handlePartition false`. If `-handlePartition false` is set, and you specify `-pinGuide true`, Trial Route ignores the `-handlePartition false` setting, and runs as if `-handlePartition true` is specified instead. If you want to use `-handlePartition false`, you must specify `-pinGuide false`.

Default: `true`

`-PKS {true | false}`

Generates routes using a Steiner routing algorithm. Net routes are used for estimating routing congestion and determining net loading and delays for timing analysis.

Default: `false`

`-printSections {true | false}`

Writes the congestion information for the chip section-by-section, to a report in the log file. The report formats the information section-by-section, with each section containing congestion information for each layer in the section.

For more information on the congestion distribution report, see "Congestion Distribution Report" in the *Encounter User Guide*.

Default: `false`

`-printWiresOnRTBlk {true | false}`

Prints the names of all wires that overlap routing blockages on the same layer to the log file. If you specify `-printWiresOnRTBlk true` and `-handlePartition true`, Trial Route also prints the names of wires that overlap temporary routing blockages derived from partition specifications.

Default: `false`

`-printWiresOnRTBlkFile {`*`fileName`*`}`

Prints the names of all wires that overlap routing blockages on the same layer to the specified report file, instead of to the log file. If you specify `-printWiresOnRTBlkFile` and `-handlePartition true`, Trial Route also prints the names of wires that overlap temporary routing blockages derived from partition specifications.

The report file contains the following wire information: the net name, the layer, and the coordinates for the section of the wire on that layer that overlaps the routing blockage. It also contains a `zoom` command for each overlapping wire section, so you can easily find it in the design. The following example shows a section of a report file:

```
net SCAN_MODE_0 wire: Layer 6 (5537.450,4726.450)-
    (5537.950,4726.750)
      zoomBox 5537.450 4726.450  5537.950 4726.750
net SCAN_MODE_0 wire: Layer 7 (5537.350,4237.350)-
     (5538.250,4727.050)
       zoomBox 5537.350 4715.600  5538.250 4727.050
net SCAN_MODE_0 wire: Layer 6 (5537.650,4237.650)-
     (5538.350,4237.950)
       zoomBox 5537.650 4237.650  5538.350 4237.950
```

Use the report file to find where wires are overlapping blockages. Wires can only overlap routing blockages when there is no other path to connect terms except to go through the blockage, or when the overlap is within a gcell that contains part of a blockage.

Default: `" "` (empty string)

`-printWiresOnRTBlkLong {true | false}`

Prints to the log file only the names of wires with an aspect ratio greater than 20 that overlap routing blockages on the same layer.

Default: `false`

`-printWiresOnRTBlkLongFile {`*`fileName`*`}`

Prints only the names of wires with an aspect ratio greater than 20 that overlap routing blockages on the same layer to the specified file, instead of to the log file.

Default: `" "` (empty string)

`-printWiresOutsideBusguide {true | false}`

Generates warning messages for all wires that are outside of the bus guides after Trial Route has completed.

Note: A wire can go outside of the specified bus guides if it is within a gcell that is at least partially covered by a bus guide. In this case, Trial Route does not generate a warning.

Default: `false`

`-ptnBdryExt` *`float`*

Specifies the spacing, in microns, between partition boundaries and wires outside of the partitions for top-level nets. Use this parameter to prevent outside wires from routing inside of partition boundaries.

Note: The `-ptnBdryExt` parameter can only be used with `-handlePartition true` and `-handlePartitionComplex true`.

Default: `0`

`-ptnBdryShr` *`float`*

Specifies the spacing, in microns, between partition boundaries and wires inside of the partitions when routing intra partition nets. Use this parameter to prevent inside wires from routing outside of partition boundaries.

Note: The `-ptnBdryShr` parameter can only be used with `-handlePartition true` and `-handlePartitionComplex true`.

Default: `0`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setTrialRouteMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-routeGuide {`*`fileName`*`}`

Specifies a route guide file to use. Guide files help control the routing topology by dividing the design into routing regions in which Trial Route then creates wires.

Guide files are generated when there are partially routed nets with `FIXED` wires in the design, and contain the following wire information: the coordinates of the wire ends, whether it is a horizontal or vertical wire, the width of the wire, and the layer number. The following example shows a section of a guide file:

```
routeGuideNet clk
wire 32.89 50 32.89 45.715 V 0.5 L 3
wire 32.89 45.715 32.89 10.045 V 0.5 L 5
wire 32.89 45.715 43.47 45.715 H 0.5 L 4
wire 32.89 10.045 7.13 10.045 H 0.5 L 4
wire 7.13 10.045 7.13 35.465 V 0.5 L 3
endRouteGuideNet
```

Default: *“ ”* (empty string)

`-routeObs {true | false}`

Allows partial routing over routing obstructions and blocks. Use this parameter when Trial Route takes a long time to complete due to blocked terminals.

Default: `false`

`-selMarkedNet {true |false}`

Routes the preselected nets first, then all remaining nets.

Default: `false`

`-selMarkedNetOnly {treu | false}`

Determines whether to route only the preselected nets.

Default: `false`

`-selNet {`*`fileName`*`}`

Routes the nets in the specified file first, then all remaining nets. The file contains a list of net names, each on a separate line. You cannot specify wild cards. Use this parameter to route a set of high priority nets first.

If you specify `-selNet` and `-minRouteLayer`, Trial Route routes the nets in the file first on the layers at and above the specified minimum routing layer, then routes the remaining nets on all layers.

If you specify `-selNet` and `-maxRouteLayer`, Trial Route routes the nets in the file first. However, all nets are routed at and below the specified maximum routing layer.

Default: `" "` (empty string)

`-selNetOnly {`*`fileName`*`}`

Routes only the nets in the specified file.

Default: `" "` (empty string)

`-skipTracks {"`*`layerName tracksToSkip`*`:`*`totalTracks`*`"}`

Skips the specified number of tracks out of the specified total number of tracks on the specified layer during routing.

For example, if you specify:

```
-skipTracks “M5 1:3”
-skipTracks “M6 1:4”
```

Trial Route skips one out of every three tracks for routing on layer *metal5* (that is, it uses two adjacent tracks, then skips the third), and skips one out of every four tracks for routing on layer *metal6* (that is, it uses three adjacent tracks, then skips the fourth).

Default: `“ ”` (empty string)

`-updateRemainTrks {true | false}`

Updates and reports congestion data after the layer assignment phase of Trial Route. Updating the congestion data displays congestion that is closer to the actual Trial Route result.

Default: `true`

`-useM1 {true | false}`

Uses layer *metal1* to route the design rather than using *metal1* for connecting to pins. Specifying this parameter provides better route congestion results.

Default: `false`

Example

- The following command writes out the congestion information section-by-section in the log file, when `trialRoute` is run:

  ```
  setTrialRouteMode -printSections true
  ```

- The following command allows partial routing over routing obstructions and blocks, when `trialRoute` is run:

  ```
  setTrialRouteMode -routeObs true
  ```

- The following command resets the `-printSections` parameter to its default value:

  ```
  setTrialRouteMode -reset -printSections
  ```

- The following command resets all `setTrialRouteMode` parameters to their default values:

```
setTrialRouteMode -reset
```

trialRoute

```
trialRoute
     [-noDetour | -highEffort]
     [-useM1]
     [-guide fileName]
     [-ignoreObstruct]
     [-handlePartition [-handleEachPartition] | -handlePartitionComplex
        [-handleEachPD]]
     [-fastRouteForPinAssign [-intraNets]]
     [-maxRouteLayer layerNumber]
     [-minRouteLayer layerNumber]
     [-noHonorNetRTLayer]
     [-selNet fileName | -selNetOnly fileName | -selMarkedNet
        | -selMarkedNetOnly]
     [-noObstruct4]
     [-floorplanMode]
     [-handlePreroute]
     [-keepExistingRoutes]
     [-PKS]
     [-noUpdateRemainTrks]
     [-ignoreDEFTrack]
     [-noPinAndObsSpacing]
     [-noSpacingTable]
     [-printSections]
     [-printWiresOnRTBlk | -printWiresOnRTBlkLong | -printWiresOnRTBlkratio
        | -printWiresOnRTBlkFile fileName | -printWiresOnRTBlkLongFile fileName]
     [-printWiresOutsideBusguide]
     [-skipTrack {layerName | all} numberOfTracksToSkip:outOfTracks]
     [-usePagedArray]
     [-ignoreGuideLayer]
     [-noConsiderMetalPinBlockage]
     [-noPinGuide]
     [-honorPin]
     [-keepEndPoints | -keepEndPoints2]
     [-blockageCostMultiple number]
     [-maxDetourRatio value]
     [-extendM1PinToM2]
     [-autoSkipTracks]
     [-ptnBdryExt value]
     [-ptnBdryShr value]
     [-ignoreAbutted2TermNet]
     [-ignorePrefExtraSpace]
     [-ignoreNetIsSpecial]
     [-ignoreRoutingHalo]
     [-handlePD [-handleEachPD] | -handlePDComplex [-handleEachPD]]
     [-PDAwareSelNet]
     [-ignoreBumpNets]
     [-ignoreRTBlockage fileName]
     [-handlePartFixedNets]
```

```
[-keepPreferredLayer]
[-routeBasedBBPin]
[-excludePartitionFile fileName]
```

Performs quick global and detailed routing for estimating routing-related congestion and capacitance values. Trial Route results are also used for pin assignment when you commit partitions in hierarchical designs.

Trial Route does not guarantee DRC-clean routing results. Do not perform signal integrity analysis on a design that has been routed using Trial Route, because these routes are only used to estimate parasitic values for timing analysis. Route designs with the NanoRoute router or the WRoute router if you want to perform signal integrity analysis.

Wire overlap can occur if a gcell is only partially covered by a routing blockage or pre-routed wires. Overlapping of nondefault rule wires can also occur during track assignment; however, the routing congestion information is correct.

You can use the `trialRoute` command after running placement. If the design uses additional placement features, such as clock tree synthesis and scan groups, use the command after running these placement features. If you have run IPO commands on your design, you must run the `trialRoute` command to update interconnections for the design.

Note: You can also set global parameters for the Trial Route program by using the `setTrialRouteMode` command. Parameters that you specify with `setTrialRouteMode` are then used automatically whenever you run Trial Route, either explicitly through the `trialRoute` command or Trial Route form, or internally during in-place optimization, partitioning, or placement.

Parameters

`-autoSkipTracks`

Classifies nets by nondefault routing rule, then routes the nets for each rule using the track constraints specified in the rule. Trial Route routes nets for the rule with the largest pitch first, then the rule with the second largest pitch, and so forth. After routing the nets for a rule, Trial Route marks them as pre-routed before routing the nets for the next rule.

Note: Wire overlap can occur if a gcell is only partially covered by a routing blockage or pre-routed wires.

`-blockageCostMultiple` *`value`*

Specifies the value by which to multiply the cost of the wire in order to avoid overlapping with a routing blockage in an area that is not completely blocked. The higher the wire number, the more Trial Routes attempts to avoid overlapping the wire with a blockage.

Use this parameter when there are long, thin routing blockages.

Value: `1-100`

`-excludePartitionFile` *`filename`*

When specified with `-handlePartitionComplex` (or `-handlePartition`), instructs Trial Route to only take into account the partitions listed in the specified file, when routing nets belonging to more than one partition. Trial Route does not have to detour around partitions not specified in the file, and they can be used for feedthrough insertion.

`-extendM1PinToM2` Reserves one *metal2* track for use by a *metal1* pin in the same gcell. When Trial Route routes the *metal1* pin, it uses that *metal2* track to route to *metal2*.

Use this parameter to achieve a more accurate model of *metal2* congestion.

`-fastRouteForPinAssign`

Creates a partition-aware routing topology for approximately 95 percent of inter-partition nets that is similar to the routing topology generated by the `-handlePartitionComplex` parameter.

Trial Route routes net groups that have a significant number of inter-partition nets in a partition-aware manner, using larger gcells. The remaining net groups with fewer inter-partition nets are routed in a manner similar to flat Trial Route. Using this method, Trial Route can generate the routing topology faster than with the `-handlePartitionComplex` parameter.

To also route intra-partition nets, specify `-intraNets` in conjunction with this parameter.

Note: The `-fastRouteForPinAssign` parameter should be used only for performing congestion-aware pin assignment.

`-floorplanMode`

Increases the size of the global routing cells with fewer iterations.

`-guide` *`fileName`*

Specifies the route guide file to use. Routing guides help control the routing topology by dividing the design into routing regions in which Trial Route then creates wires.

Routing guide files contain the following wire information: the coordinates of the wire ends, whether it is a horizontal or vertical wire, the width of the wire, and the layer number. The following example shows a section of a routing guide file:

```
routeGuideNet clk
wire 32.89 50 32.89 45.715 V 0.5 L 3
wire 32.89 45.715 32.89 10.045 V 0.5 L 5
wire 32.89 45.715 43.47 45.715 H 0.5 L 4
wire 32.89 10.045 7.13 10.045 H 0.5 L 4
wire 7.13 10.045 7.13 35.465 V 0.5 L 3
endRouteGuideNet
```

Note: Trial Route honors partially-routed nets with wires marked `+ FIXED`, if you specify the `-handlePreroute` parameter.

`-handleEachPartition`

When specified with `-handlePartitionComplex`, routes each intra-partition net set one at a time in order to avoid intra-partition nets being routed through other partitions that are located inside.

When two partitions abut each other or are too close, a routing blockage cannot be generated. As a result, an intra-partition net can be routed through another partition located inside of the partition. Use this parameter to block the net from crossing the boundary of the other partition.

This parameter can only specified with the `-handlePartition` or `-handlePartitionComplex` parameters.

`-handleEachPD`

When specified with `-handlePDComplex`, routes each intra-power domain net set one at a time in order to avoid intra-power domain nets being routed through other power domains that are located inside.

When two power domains abut each other or are too close, a routing blockage cannot be generated. As a result, an intra-power domain net can be routed through another power domain located inside of the power domain. Use this parameter to block the net from crossing the boundary of the other power domain.

Note: This parameter can only specified with the `-handlePD` or `-handlePDComplex` parameters.

`-handlePartFixedNets`

Re-routes partially routed nets with `FIXED` or `COVER` wires.

If your DEF file contains partially-routed nets with `FIXED` or `COVER` wires, Trial Route completes the routing for the nets. Trial Route cannot maintain the exact location of the wires; however, they are accurate within the gcell area. Also, if you write out a DEF file after routing, all wires for the net are marked `FIXED`. You can no longer identify the original `FIXED` wires.

Trial Route does not re-route completely routed nets with `FIXED` or `COVER` wires.

Default: Maintains all partially and completely routed nets with `FIXED` or `COVER` wires

`-handlePartition`

Routes intra-partition and global nets using the specifications of the partitions, such as area and the number of layers. This routing is based on the resources available after the chip is partitioned.

Use this parameter to gauge the congestion at the full chip level for the chip to be partitioned.

Note: For flat partitioned designs, you can specify `-handlePartition` to simulate channel-based routing. For channelless designs, you must perform feedthrough insertion before using this parameter.

`-handlePartitionComplex`

Routes a net belonging to more than one partition so that the routing does not violate partition boundaries, in addition to routing intra-partition and global nets using the specifications of the partitions. For example, if a net belongs to partitions `P1` and `P3`, but not `P2`, the net is routed without going over the `P2` partition boundary.

Note: For flat partitioned designs, you can specify `-handlePartitionComplex` to simulate channel-based routing. For channelless designs, you must perform feedthrough insertion before using this parameter.

`-handlePD`

Routes intra-power domain and global nets using the specifications of the power domains. Trial Route routes intra-power domain nets only within their power domains. Global nets that do not connect with a power domain do not get routed through that power domain.

`-handlePDComplex`

Routes nets belonging to more than one power domain so that the routing does not violate power domain boundaries, in addition to routing intra-power domain and global nets using the specifications of the power domains.

For example, if a net belongs to power domains `PD1` and `PD3`, but not `PD2`, Trial Route routes the net without going over the `PD2` power domain boundary.

`-handlePreroute`

Does not re-route nets with a database status of `IsIgnoreInRoute`, but re-routes all other nets.

You can use the `dbIsNetIgnoreInRoute` database command to check if a net has a database status of `IsIgnoreInRoute`. To set the status of a net to `IsIgnoreInRoute`, use the `dbSetIsNetIgnoreInRoute` database command.

Note: Trial Route automatically maintains completely routed nets marked as `FIXED` or `COVER`.

`-highEffort`

Runs additional iterations for resolving difficult routing congestions.

`-honorPin` Honors pre-assigned pins (pins marked `FIXED`) and assigned pins (pins marked `PLACED`). If these pins exist, Trial Route automatically calls `-handlePartition` to keep wires within or outside partition boundaries. If there are no existing pre-assigned or assigned pins, Trial Route runs once, which is its default behavior.

`-ignoreAbutted2TermNet`

Does not route nets that have two pins on the same layer whose pin geometries abut (that is, touch each other).

`-ignoreBumpNets` Does not route nets that are connected to bumps. Nets with connections to bumps are typically routed using the `fcroute` command. Use this parameter to prevent Trial Route from routing these nets.

`-ignoreDEFTrack` Ignores the track definitions specified in the `TRACKS` statement of the DEF file and defines its own tracks. Improperly defined tracks can cause Trial Route to take a long time to complete, and to sometimes crash.

Default: Uses the tracks defined in the DEF file.

`-ignoreGuideLayer` Ignores the layer information in a route guide file, if one is specified, and determines the wire layers during routing.

Default: Maintains the wire layers specified in the route guide file during routing. The wire layer information specified in the route guide file includes the coordinates of the wire ends, whether it is a horizontal or vertical wire, the width of the wire, and the layer number.

`-ignoreNetIsSpecial` Ignores nets that have a status of Special in the database. (You can use the `dbIsNetSpecial` database command to check if a net has a database status of Special.) Trial Route does not route special nets, but considers existing special wires when calculating congestion.

`-ignoreObstruct` Routes over all blocks and partitions.

Default: Only routes over blocks and partitions that do not have routing blockages specified in their LEF files. Routing blockages for partitions are specified in a LEF file created by the Encounter software. This LEF file contains information set with the Specify Partition form, and is created when the partition is saved either with the Save Partition form or with the `savePartition` command.

`-ignorePrefExtraSpace`

Ignores the extra net spacing specified for individual nets with the `setAttribute` and `setCTSMode` commands.

`-ignoreRoutingHalo` Does not limit routing inside of routing halos.

Default: Trial Route considers it high cost to route inside a routing halo, and only routes inside the halo area to connect to pins on the boundary of the block.

`-ignoreRTBlockage` *`fileName`*

Ignores routing blockages within a specified area to avoid creating detours around them. When you specify `-ignoreRTBlockage`, Trial Route reads the specified file, and ignores all routing blockages and cell obstructs that are within the box areas defined in the file.

Use the following format to define the box areas in the file:

`layername` (*`llx`*,*`lly urx`*,*`ury`*)

For example:

```
M3 (288.431,279.117 377.079,648.289)
```

Default: Does not ignore the routing blockages

`-intraNets` When specified with `-fastRouteForPinAssign`, routes intra-partition nets, in addition to routing inter-partition and top-level nets.

Note: This parameter must be specified in conjunction with the `-fastRouteForPinAssign` parameter. The `-fastRouteForPinAssign` and `-intraNets` parameters should be used only for performing congestion-aware pin assignment.

`-keepEndPoints`

Maintains only the floating wires marked with an “F” in the guide file, and completes their routing. All other floating guide files wires are deleted.

For example, the `-keepEndPoints` parameter preserves the following floating guide file wire:

```
wire 3819.3 -289.5 3827.1 -289.5 H 0.5 L 2 F
```

Currently, you must manually add the “F” in the guide file to mark the floating wires that you want to preserve.

If you specify this parameter, you must also specify the `-guide` parameter.

`-keepEndPoints2`

Maintains all floating wires in the guide file and completes their routing.

`-keepExistingRoutes`

Maintains all existing pre-routed nets.

Note: Trial Route automatically maintains partially and completely routed nets marked as `FIXED` or `COVER`.

`-keepPreferredLayer`

Attempts to route short pin connections from long wires by routing in the preferred layer direction on layers. For example:

M3(H) -> M2(V) -> M1(pin)

If you do not specify this parameter, Trial Route can create the connections by routing in both the horizontal and vertical directions on a layer. For example:

M2(H) -> M2(V) -> M1(pin)

`-maxDetourRatio` *`value`*

Attempts to re-route two-pin nets to reduce wire length when the wire length is greater than:

value x half the net bounding box perimeter

Trial Route reduces the wire length by relaxing the cost for congested gcells; therefore, reducing wire length can increase congestion.

Use this parameter when there are two-pin nets with long detours that slow timing.

Type: Float

Note: This parameter applies only to two-pin nets.

`-maxRouteLayer` *`layerNumber`*

Limits routing to layers at and below the specified layer. All metal layers above the specified metal number are not routed.

Default: Top-most layer

Note: The value you specify must be less than or equal to any global maximum routing layer limit set with the `setMaxRouteLayer` command. If you specify a value that is greater than the global value, Trial Route automatically resets the value to be equal to the global value.

`-minRouteLayer` *`layerNumber`*

Limits routing to layers at and above the specified layer. However, layers below the specified layer can still be used for routing if necessary, such as for short connections to pins.

You must specify a number greater than 1. Specify the `-useM1` parameter to use layer *metal1* for routing.

Default: 2

Note: If you specify this parameter and `-selNet`, Trial Route routes the nets in the file first on the layers at and above the specified minimum routing layer, then routes the remaining nets on all layers.

`-noConsiderMetalPinBlockage`

Does not consider pins on layers *metal2* and higher as routing blockages.

Default: Considers pins on layers *metal2* and higher as routing blockages, and reduces the routing resources in those areas.

`-noDetour`

Completes a quick routing through all congested areas. Use this parameter only when routing a poorly designed netlist in which a part of the design is good and needs to be analyzed.

Note: You should not use this parameter for a production design. The routing of the design is not accurate because some nets are routed through congested areas or over blockages in order to complete the routing.

`-noHonorNetRTLayer`	Does not honor the maximum and minimum routing layer limits specified for individual nets with the `setAttribute` command.
`-noObstruct4`	Allows partial routing over routing obstructions and blocks. You can use this parameter when trial routing takes a long time to complete due to blocked terminals.
`-noPinAndObsSpacing`	Ignores obstruction and pin port spacing and width definitions if specified in the `MACRO` section of the LEF file.
`-noPinGuide`	Does not use the pin guide statements in the floorplan file to guide the routing through partition pin points. *Default:* Uses the pin guide statements in the floorplan file, if they exist. If pin guide statements exist, Trial Route automatically calls `-handlePartition` to keep wires within or outside partition boundaries. If there are no existing pin guide statements, Trial Route runs once, which is its default behavior.
`-noSpacingTable`	Ignores spacing table rules if specified in the `LAYER` section of the LEF file.
`-noUpdateRemainTrks`	Does not update congestion data after the layer assignment phase of Trial Route. The congestion that is displayed is less accurate. *Default:* Updates congestion data after the layer assignment phase of Trial Route mode. This displays congestion that is closer to the actual Trial Route result.
`-PDAwareSelNet`	When specified with `-handlePD` and `-selNet` *`filename`*, routes only the intra-power domain nets that are specified in *`filename`* within their power domains. Global nets that do not connect with a power domain are then routed outside of the power domains. **Note:** This parameter must be specified in conjunction with the `-handlePD` and `-selNet` parameters.
`-PKS`	Generates net routes using a Steiner routing algorithm. These net routes are used for estimating routing congestion and determining net loading and delays for timing analysis.

`-printSections`	Writes the congestion information for the chip section-by-section, to a report in the log file. If you specify this parameter, the report formats the information section-by-section, with each section containing congestion information for each layer in the section. *Default:* Writes congestion information to a report in the log file for the entire chip. For more information on the congestion distribution report, see "Congestion Distribution Report," in the *Encounter User Guide*.
`-printWiresOnRTBlk`	Prints the names of all wires that overlap routing blockages on the same layer. When used with the `-handlePartition` parameter, Trial Route also prints the names of wires that overlap temporary routing blockages derived from partition specifications.

`-printWiresOnRTBlkFile` *`fileName`*

Prints the names of all wires that overlap routing blockages on the same layer to the specified file, instead of to the log file. When used with the `-handlePartition` parameter, Trial Route also prints the names of wires that overlap temporary routing blockages derived from partition specifications.

The report file contains the following wire information: the net name, the layer, and the coordinates for the section of the wire on that layer that overlaps the routing blockage. It also contains a `zoom` command for each overlapping wire section, so you can easily find it in the design. The following example shows a section of a report file:

```
net SCAN_MODE_0 wire: Layer 6 (5537.450,4726.450)-
    (5537.950,4726.750)
      zoomBox 5537.450 4726.450  5537.950 4726.750
net SCAN_MODE_0 wire: Layer 7 (5537.350,4237.350)-
     (5538.250,4727.050)
       zoomBox 5537.350 4715.600  5538.250 4727.050
net SCAN_MODE_0 wire: Layer 6 (5537.650,4237.650)-
     (5538.350,4237.950)
       zoomBox 5537.650 4237.650  5538.350 4237.950
```

You can use the report file to find where wires are overlapping blockages. Wires can only overlap routing blockages when there is no other path to connect terms except to go through the blockage, or when the overlap is within a gcell that contains part of a blockage.

`-printWiresOnRTBlkLong`*`ratio`*

Prints only the names of wires with an aspect ratio greater than the specified value that overlap routing blockages on the same layer.

Default: `20`

`-printWiresOnRTBlkLong`

Prints only the names of wires with an aspect ratio greater than 20 that overlap routing blockages on the same layer.

`-printWiresOnRTBlkLongFile` *`fileName`*

Prints only the names of wires with an aspect ratio greater than 20 that overlap routing blockages on the same layer to the specified file, instead of to the log file.

`-printWiresOutsideBusguide`

Generates warning messages for all wires that are outside of the bus guides after Trial Route has completed.

Note: A wire can go outside of the specified bus guides if it is within a gcell that is at least partially covered by a bus guide. In this case, Trial Route does not generate a warning.

`-ptnBdryExt` *`value`*

Specifies the spacing, in microns, between partition boundaries and wires outside of the partitions for top-level nets. Use this parameter to prevent outside wires from routing inside of partition boundaries.

Note: This parameter can only be used with the `-handlePartition` and `-handlePartitionComplex` parameters.

`-ptnBdryShr` *`value`*

Specifies the spacing, in microns, between partition boundaries and wires inside of the partitions when routing intra partition nets. Use this parameter to prevent inside wires from routing outside of partition boundaries.

Note: This parameter can only be used with the `-handlePartition` and `-handlePartitionComplex` parameters.

`-routeBasedBBPin`

Determines near-optimal locations for blackbox pins taking into account top-channel congestion, and places blackbox pins at these locations. Trial Route then creates routes to the blackbox pins without crossing over the blackboxes.

`-selMarkedNet`

Routes the preselected nets first, then all remaining nets.

`-selMarkedNetOnly`

Routes only the preselected nets.

`-selNet` *`fileName`*

Routes the nets in the specified file first, then all remaining nets. The file contains a list of net names, each on a separate line. You cannot specify wild cards. Use this parameter to route a set of high priority nets first.

Note: If you specify this parameter and `-minRouteLayer`, Trial Route routes the nets in the file first on the layers at and above the specified minimum routing layer, then routes the remaining nets on all layers.

If you specify this parameter and `-maxRouteLayer`, Trial Route routes the nets in the file first. However, all nets are routed at and below the specified maximum routing layer.

`-selNetOnly` *`fileName`*

Routes only the nets in the specified file.

`-skipTrack {`*`layerName`*` | all}` *`numberOfTracksToSkip`*`:`*`outOfTracks`*

Skips the specified number of tracks out of the specified total number of tracks during routing. You can skip tracks on a specified layer, or on all layers. For example, if you specify:

```
trialRoute -skipTrack M5 1:3 -skipTrack M6 1:4
```

Trial Route skips one out of every three tracks for routing on layer *metal5* (that is, it uses two adjacent tracks, then skips the third), and skips one out of every four tracks for routing on layer *metal6* (that is, it uses three adjacent tracks, then skips the fourth).

`-useM1`

Uses layer *metal1* to route the design rather than using *metal1* for connecting to pins. This provides better route congestion results.

Default: Uses *metal1* to route to pins.

`-usePagedArray`

Breaks large linear arrays into chunks, or paged arrays for better memory usage.

You should only use this parameter for large jobs (that is, jobs with approximately one million cells or more) that are running out of memory. This feature works best when Trial Route is run in default mode.

Example

The following command runs the trial route program in a medium effort level with assigned preroutes from the `clocktree.guide` file.

```
trialRoute -guide clocktree.guide
```

Related Topics

- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

wroute

```
wroute
     [-mode {globalOnly | globalAndFinal | incrFinal | incrGlobalAndFinal
        | postRouteRepair | incrPostRouteRepair}]
     [-area {x1 y1 x2 y2}]
     [-timingDriven
        [-timingOptimize]
        [-extraGlobalConfig fileName]]
     [-wdbName fileName]
     [-multiCpu number]
     [-maxRuntime minutes]
     [-saveInterval minutes]
     [-noAutoStop]
     [-nets {all | namedFirst -namedNets {list_of_nets}
        | namedOnly -namedNets {list_of_nets}}]
     [-straightenNets {list_of_nets}]
     [-pinNoAllowShorts]
     [-pinEncloseWire]
     [-pinOnGridOnly]
     [-antennaFixPassNumber passNumber]
     [-antennaInsertCellPassNumber passNumber]
     [-antennaTopLayerLimit layerNumber]
     [-topLayerLimit layerNumber]
     [-relaxTopLayerLimit]
     [-bottomLayerLimit layerNumber]
     [-namedTopLayerLimit layerNumber -namedNets {list_of_nets}]
     [-namedBottomLayerLimit layerNumber -namedNets {list_of_nets}]
     [-noSearchAndRepair]
     [-fullSearchAndRepair]
     [-modifyPreroutePass passNumber]
     [-taperMaxDistance taperDistance]
     [-taperPinSelection {ruleBased | inputOnly | all}]
     [-xtalkRule {prevention | ruleSpec}]
     [-xtalkRuleThreshold thresholdDistance]
     [-xtalkSegmentThreshold thresholdDistance]
     [-xtalkFixNetFile fileName]
     [-optimizeMode {viasOnlyDuring | viasOnlyAfter | allAfter
        | forceViasOnlyAfter | forceAllAfter}]
     [-grouteGrid {existing | blockAligned | uniform}]
     [-blockMinimizePlanarRouting]
     [-mixedBlockAndCellMode {automatic | block | cell}]
     [-routeToCenterOfSpecialNet]
     [-viaOnGridBelowLayer layerNumber]
     [-extraConfig fileName]
```

Routes placed signal and clock nets in block-and-cell designs using an area-based algorithm.

Before routing standard cells with `wroute`, complete the following steps:

1. Place the design (including placement of filler cells).
2. Preroute special nets.

Parameters

`-antennaFixPassNumber` *`passNumber`*

Checks and repairs process antenna violations during search-and-repair routing by automatically shortening any wire whose area exceeds the gate/wire area ratio set in the LEF file. The router corrects design rule violations before it corrects process antenna violations. When the number of design rule violations is comparable to the number of process antenna violations, the router starts correcting process antenna violations.

Do not change the setting for this parameter unless you want to disable the antenna check. This parameter does not guarantee that the router will resolve all antenna violations. If the routing area is congested, process antenna violations can occur, just as shorts and spacing violations can occur.

Default: If you do not specify this parameter, the router determines automatically when to start correcting process antenna violations.

Set this parameter to `0` to disable the antenna check.

Note: If you specify `-optimizeMode`, the router does not start repairing process antenna violations until after running optimization. For more information, see `-optimizeMode`.

`-antennaInsertCellPassNumber` *`passNumber`*

Adds antenna diode cells to repair process antenna violations at or after the specified search-and-repair pass. Generally, the router attempts to correct process antenna violations by shortening wires before it starts to insert antenna cells. To disable this parameter, specify `0`.

Default: If you do not specify this parameter, the router determines automatically when to start inserting antenna cells.

`-antennaTopLayerLimit` *`layerNumber`*

Starts adding antenna diode cells if process antenna violations remain after layer-hopping reaches the specified layer. Use this parameter to minimize congestion. For example, in a design with six or more layers, if you set the value of this parameter to `4`, `wroute` attempts to repair antenna violations by layer-hopping until it reaches *metal4*. Then it starts repairing the violations by adding antenna diode cells.

Default: `0` (disabled)

`-area {`*`x1 y1 x2 y2`*`}`

Specifies the area to route. You must use curly braces to enclose the `x` and `y` values.

Default: **`{`**`0 0 0 0`**`}`** (entire core area)

`-blockMinimizePlanarRouting`

Penalizes routing over blocks when only one routing layer is available. Routing on one layer increases the possibility that wires will cross.

`-bottomLayerLimit` *`layerNumber`*

Allows routing of regular nets below the specified layer, but at a very high cost. This limits the wire length below the specified layer.

Default: (no limit)

`-extraConfig` *`fileName`*

Specifies an additional configuration file for routing. You can use the additional configuration file to set `wroute` parameters that are not available within Encounter. If there is a discrepancy between the parameters in the extra configuration file and the Encounter configuration file, `wroute` uses the values in the extra configuration file.

Default: `""`

Note: For information on the additional parameters you can set, see the *Ultra Router Reference*.

`-extraGlobalConfig` *`fileName`*

Specifies an extra configuration file to use during PKS-based global routing.

Note: You must specify `-timingDriven` in order to use the `-extraGlobalConfig` parameter.

`-fullSearchAndRepair`

Runs full search-and-repair routing.

Default: Runs WRoute in quick mode, by turning off certain time consuming options and limiting the number of search-and-repair passes.

`-grouteGrid {existing | blockAligned | uniform}`

Sets the global routing grid generation mode. The global routing grid partitions the routing portion of a standard cell design into rectangles called global routing cells (gcells).

Default: `uniform`

`existing`	Uses the global routing grid defined in the DEF file.
`blockAligned`	Generates a global routing grid that aligns with the channels and macro block boundaries in your design. This parameter improves routing quality for block-based designs because, by aligning the grids, the global router can route more effectively in empty gcells that exist in the channel area.
`uniform`	Generates a uniform global routing grid.

`-maxRuntime` *`minutes`*

Specifies the maximum run time for the router. This parameter is useful when you want to ensure that the router stops after an amount of time you specify, or if you want to look at parameter settings and route a specific area, to see if it is easily routable, but you do not want to optimize the routing.

Default: `1440`

`-mixedBlockAndCellMode {automatic | block | cell}`

Specifies whether to scan the design to determine the necessity for chip-assembly routing. Changes heuristics during detailed routing, depending on the percentage of the design area that is covered by blocks, rings, or pads. In chip-assembly mode, the router runs an order of magnitude faster than in non-chip-assembly mode at the expense of slightly degraded routing quality.

Default: `automatic`

Important

Cadence recommends that you change the setting of this parameter only when you are certain that you need to override the default.

`automatic` Lets the router determine when to route in chip-assembly mode.

`block` Routes in chip-assembly mode only.

`cell` Routes in non-chip-assembly mode only.

`-mode {globalOnly | globalAndFinal | incrFinal | incrGlobalAndFinal | postRouteRepair | incrPostRouteRepair}`

Specifies whether the router runs global, final, or search-and-repair routing, or a combination of these.

Default: `globalAndFinal`

`globalOnly` Runs global routing.

During global routing, the router interconnects the regular (signal) nets defined in the NETS section of the DEF file for the design, based on the availability of routing tracks. The router finds generalized pathways, without laying down actual wires, and makes iterative passes to optimize the global routing, shorten wire length, and minimize the use of vias.

`globalAndFinal`

Runs global and detailed routing.

During detailed routing, the router follows the global routing plan and lays down actual wires that connect the pins to their corresponding nets. The primary goal of detailed routing is to complete all of the required interconnects without creating shorts or spacing violations. Unlike the global router, however, the detailed router is not conservative. It will create shorts or spacing violations rather than leave unconnected nets. Detailed routing stops automatically if it exceeds the time limit you set for routing or if it cannot make further progress on routing the design.

`incrFinal` Runs detailed routing on a database that has already had at least one global routing pass.

Note: If you specify this value, you must also specify the `-wdbName` parameter.

`incrGlobalAndFinal`

Runs both global and detailed routing, including the search-and-repair step, on a previously routed database. This parameter is useful if your previous detailed route could not remove all violations.

Note: If you specify this value, you must also specify the `-wdbName` parameter.

`incrPostRouteRepair`

Reads the netlist and routing from the existing routed database (`.wdb`), then updates WRoute's connectivity from the data in the Encounter software, ignoring any routing in the data. The router then incrementally fixes any routing violations and routes any new nets.

You typically use this option when gate-resizing buffer insertion and deletion is done during postroute repair for timing or signal integrity violations.

Note: If you specify this option, you must also specify the `-wdbName` parameter.

`postRouteRepair`

Runs both final and global routing during postroute optimization. Makes the router more error tolerant by allowing it to delete problematic nets, such as nets with loops and nets for which it cannot extract connectivity.

`-modifyPreroutePass` *`passNumber`*

Specifies the search-and-repair pass when the router can begin modifying prerouted regular nets. The router does not modify nets whose location is fixed.

Default: 2

`-multiCpu` *`number`*

Specifies the number of processors inside a single workstation that you can use in routing. WRoute can use up to 4 processors during global routing and 16 processors during detailed routing. If you enter a number greater than 16, or you do not have the appropriate number of Route Accelerator licenses, the router outputs a warning message and resets the number of processors to a legal number.

Default: 1

Note: You must have a Route Accelerator license for each additional processor you use for multi-threading.

`-namedBottomLayerLimit` *`layerNumber`*

Specifies the bottom layer for routing nets specified by `-namedNets`. You can specify a bottom layer only, or both a top layer and a bottom layer.

Default: `0` (no limit)

`-namedNets {`*`list_of_nets`*`}`

Specifies the subset of nets if you are routing only a subset of the nets in the design, or are routing a subset of nets before routing other nets.

Default: “”

You can specify net names using the following methods:

`~` Routes all nets *except* the specified nets.

`netName` Specifies the name of a net to route.

`“list_of_nets”`

Lists the DEF names of the nets to route. The list has a 4,000-character limit. Enclose the list in quotation marks. Use the following to specify multiple nets:

- `“@clock”` selects all nets marked with `+ USE CLOCK`.
- `“@subnet”` selects all nets with subnets.
- `“@rule”` selects all nets using a nondefault rule.
- `“@rule:rulename”` selects all nets using the specified nondefault rule (rulename).
- `“@shield”` selects all shielded nets.
- `“@power”` selects all nets marked with `+ USE POWER`.
- `“@ground”` selects all nets marked marked `+ USE GROUND`.

"< fileName"	Specifies a file that contains the list of names of all the nets to route, with one name on each line. You *cannot* use regular expressions with the <.

`-namedTopLayerLimit` *`layerNumber`*

Specifies the top layer for routing nets specified by `-namedNets`. You can specify a bottom layer only, or both a top layer and a bottom layer.

Default: `0` (no limit)

`-nets {all | namedOnly | namedFirst}`

Specifies whether `wroute` routes all the regular nets in the design, a subset of the nets, or routes some nets before others.

Default: `all`

`all`	Routes all the regular nets in the design.
`namedOnly`	Routes only the nets specified by `-namedNets`.
`namedFirst`	Routes the nets specified by `-namedNets`, then routes the rest of the nets in the design.

`-noAutoStop` Forces the router to continue detailed routing, even if the router encounters a large number of violations.

If you omit this parameter, and if the router finds too many violations, it completes global routing, makes an initial pass at detailed routing, saves the data, generates a warning message, and stops.

`-noSearchAndRepair` Prevents the router from running search-and-repair routing after the initial detailed routing.

During search and repair, the router creates switch boxes centered on design rule violations, then reroutes each switch box to eliminate the violation. Sometimes there are more violations after search and repair, but usually the new violations are easier to correct. Search and repair propagates shorts toward less congested parts of the design by replacing difficult shorts with others on the boundaries of switch boxes, where they are easier to resolve.

```
-optimizeMode {viasOnlyDuring | viasOnlyAfter | allAfter
    | forceViasOnlyAfter | forceAllAfter}
```

Specifies whether and when to minimize wire length and the number of vias.

`viasOnlyDuring`

Replaces single-cut vias with double-cut vias during both the initial and search-and-repair phases of final routing. Wroute attempts the replacement even if the design has violations.

`viasOnlyAfter`

Replaces single-cut vias with double-cut vias after detailed routing is complete and violation free.

`allAfter` Minimizes wire length and the number of vias after detailed routing is complete and free of design rule violations. The router then attempts to remove extra vias by jogging on wires in the nonpreferred direction. This blocks tracks but does not cause problems because routing is already complete.

Do not set this parameter if your design has design-rule violations. If you use this parameter, the router does not optimize the design.

Note: If you specify this parameter, the router does not attempt to repair process antenna violations until after optimization.

`forceViasOnlyAfter`

Replaces single-cut vias with double-cut vias after detailed routing is complete, even if the design has violations.

`forceAllAfter`

Minimizes wire length and the number of vias after detailed routing is complete, even if the design has violations.

`-pinEncloseWire` Forces the router to connect to a pin if the pin encloses a wire, or to connect to the center of a wire if the wire is wider than the pin.

`-pinNoAllowShorts`

Prevents the router from shorting different ports of the same pin. A pin can have multiple ports, as defined in the LEF file. When multiple ports are defined for a pin, the router chooses one port and makes all connections to it.

When you set this parameter, the router does not short other geometries in the chosen port or other ports defined for the pin. Thus, it is slightly more difficult for the router to make connections because it has fewer choices.

If you omit this parameter, the router might generate wires that short geometries in the chosen port to other geometries in other ports on the same pin.

`-pinOnGridOnly` Forces the router to route to on-grid geometries of the port only. Use this parameter when all the design components are on-grid and the router will not route off-grid.

`-relaxTopLayerLimit`

Allows routing on layers higher than the one specified by `-topLayerLimit`. When you set this parameter, the router might route to a layer higher than the one set by `-topLayerLimit`, but at a high cost.

`-routeToCenterOfSpecialNet`

Forces connections to the center of wide stripes only, not to power or ground pins. Use this parameter to force the router to connect tie-high nets and tie-low nets to stripes.

`-saveInterval` *`minutes`*

Specifies the interval for automatically saving the routing data to disk.

Default: `120`

`-straightenNets {`*`list_of_nets`*`}`

Deletes specified nets in a fully routed design and reroutes them as straight as possible.

Default: `""`

`-taperMaxDistance` *`taperDistance`*

Specifies the maximum allowable taper distance, starting from the bounding box of a pin, in micrometers. To disable automatic tapering, set this parameter to `0`.

Default: If you do not specify this parameter, the router automatically determines the optimal taper distance.

Note: This parameter does not force the router to taper pins—it merely sets the maximum distance for tapering. Do not set this parameter except to disable automatic tapering.

`-taperPinSelection {ruleBased | all | inputOnly}`

Specifies which wires the router tapers.

Default: `ruleBased`

Note: This parameter lets the router taper wires—it does not tell the router when to taper the wires. The router might taper wires because of violations, congestion, or other reasons.

`ruleBased` Tapers wires based on the nondefault spacing rules defined in the technology library. You can specify several wire widths and associate special wire spacing with the widths.

- If the spacing associated with a wide wire is the same as for a regular wire, the router assumes that the wire is widened to prevent wire self-heat effects. In this case, the router tapers wide wires at input pins. (The behavior is like `inputOnly`.)
- If the spacing associated with a wide wire is larger than for a regular wire, the router assumes that the wider spacing is used to control crosstalk effects. In this case, the router tapers wires at all pins where violations occur. (The behavior is like `all`.)

`inputOnly`	Tapers wires at input pins if violations occur. Does not taper wide wires at the output or in-out pins.
`all`	Tapers wires at all pins where violations occur.

`-timingDriven`

When specified with `-mode globalOnly`, runs PKS-based global route and saves the results in the database (`.wdb`).

When specified with `-mode globalAndFinal`, runs PKS-based global route, saves the results in the database, then runs WRoute for final routing.

Note: If you specify this parameter with `-mode globalOnly`, you must then run final routing (`-mode incrfinal`) to finish the route. If you decide to perform final routing in a new session in the Encounter software, you must use the `restoreDesign` command to load the correct routing data into the design. This data can be found in the top module design file *`top`*`_tdr.enc.dat`.

`-timingOptimize`

When specified with `-mode globalOnly`, runs PKS-based global route, PKS-based optimization, then PKS-based global route again, and saves the results in the database (`.wdb`).

When specified with `-mode globalAndFinal`, runs the same procedures as described above, then runs WRoute for final routing.

Note: You must specify `-timingDriven` in order to use the `-timingOptimize` parameter.

`-topLayerLimit` *`layerNumber`*

Limits the top routing layer to the layer you specify. This parameter is useful if you want to reserve the top routing layers for power or ground or other special net routing.

Default: `0` (no limit)

`-viaOnGridBelowLayer` *`layerNumber`*

Permits on-grid vias below the specified routing layer.

Default: `0` (no layer specified)

`-wdbName` *`fileName`*

Specifies the name of the current `wroute` database.

Default: `design_name``.wdb`

`-xtalkFixNetFile` *`fileName`*

Uses the specified file to repair crosstalk violations after routing by pushing adjacent nets aside. Use this parameter on a fully routed database only. You can use the Celtic™ crosstalk analyzer or another crosstalk analyzer to generate a list of nets with crosstalk problems to use as input.

Default: `""`

`-xtalkRule {prevention | ruleSpec}`

Specifies the interaction rules for crosstalk classes. The rules apply to regular and special wires, but not to pins or obstructions.

Note: To use interaction rules, you must specify `+ XTALK` *`class`* as a net attribute in your DEF file. The default class for any net without a class number is `0`.

Class interactions are symmetric. For example, if *`class_a`* repels *`class_b`*, then *`class_b`* repels *`class_a`*. Class interactions might add slight routing penalties to your design.

Default: `""`

Note: To minimize crosstalk, Cadence recommends you specify `xtalkRule prevention` or `xtalkRule "0+0"`.

`prevention` — Spaces out wires over the blocks by one track if possible. Specify `xtalkRule prevention` `xtalkRule "0+0"` during chip-assembly or mixed chip-assembly and block routing to prevent crosstalk.

Use the following syntax in the class interaction rule.

`class` — Specifies the crosstalk class.

Note: If the class is `0` (the default), `XTALK` will not be written to the DEF file.

`"` — Specifies the beginning or the end of the string that defines the class interaction rule.

`+` — Specifies repulsion. This is the class interaction operator.

`|` — Separates the classes.

`,` Separates class groups.

Example Rule `"1|2|3+2|3, 4+0"`

In the example,

- Nets in classes 1, 2, and 3 repel nets in classes 2 and 3.
- Nets in classes 2 and 3 repel nets in classes 1, 2, and 3.
- Nets in class 4 repel nets in class 0.
- Nets in class 0 repel nets in class 4.

`-xtalkRuleThreshold` *`thresholdDistance`*

Specifies a wire-length threshold in micrometers for nets that are affected by `-xtalkRule`.

- The router considers nets whose estimated wire length after global routing is equal to or greater than the threshold as active nets for class interaction.
- The router considers nets whose estimated wire length after global routing is less than the threshold as inactive nets and removes their class assignments.

Note: To use `xtalkRuleThreshold`, you must specify `+ XTALK` *`class`* as a net attribute in your DEF file. The default class for any net without a class number is `0`.

Default: `100` µm

`-xtalkSegmentThreshold` *`thresholdDistance`*

Specifies the maximum allowable length in micrometers for parallel wire segments. If the router finds parallel wire segments that exceed the specified length, it attempts to space out the wires by one additional track within a global routing cell (gcell) to prevent crosstalk. If the gcell does not have enough room, the router completes the routing without the additional spacing.

Default: `50` µm

Example

The following command runs global and detailed routing with WRoute and applies crosstalk rules to repair violations:

```
wroute -mode globalAndFinal -optimize -xtalkRule "1+0|1" \
   -xtalkRuleThreshold 50
```

In this example, WRoute applies the following crosstalk rules to separate nets with `+ XTALK 0` and `+ XTALK 1` in the DEF file:

- Nets in class 1 repel nets in class 0.
- Nets in class 0 repel nets in class 1.
- Nets in class 1 repel other nets in class 1.

WRoute applies the rules to nets longer than the specified threshold. It measures the wire length of the nets after global routing.

Note: The units for the threshold are in microns.

For more information on `wroute`, including instructions on running it in standalone mode, see the *Ultra Router Reference*.

30

Timing Constraint Commands

create_clock

```
create_clock
     -period period_value
     [-name clock_name]
     [-waveform edge_list]
     [-add]
     [sources]
```

Creates a clock object and defines its waveform in the current design. If you do not specify any sources, but you specify the `-name` *`clock_name`* argument, a virtual clock is created. The new clock has an ideal clock latency, and the software does not assume any propagation delay through the clock network.

The clock that you create using the `create_clock` command cannot have multiple pulses per period. You can specify multiple constraints on the same pin using the `-add` parameter.

If one of the sources is already the source of a clock, the source is removed from that clock. If you need to specify multiple clocks on the same source for simultaneous analysis, you must specify the `-add` parameter, to add this clock to the existing clock.

For example, if you specify the following command sequence, the PH2 clock waveform overwrites the PH1 waveform.:

```
create_clock -name PH1 ... [get_ports sysclk]
create_clock -name PH2 ... [get_ports sysclk]
```

To have both clocks applied, you must specify the following command sequence:

```
create_clock -name PH1 ... [get_ports sysclk]
create_clock -name PH2 -add ... [get_ports sysclk]
```

You can create a collection of clocks, which is a group of objects referenced by a string identifier using the `get clocks` command.

Parameters

`-add`	Specifies multiple clocks on the same source for simultaneous analysis with different clock waveforms. You must specify the `-name` parameter with this parameter.

`-name` *`clock_name`*
: Specifies the string name of the ideal clock being created. If you do not specify this option, the clock gets the same name as the first clock source that you specify using the `-sources` option. If you do not specify any sources, you must specify the `-name` *`clock_name`* option, which creates a virtual clock not associated with a port or pin.

`-period` *`period_value`*
: Specifies the value of the ideal clock in library time units.

`sources`
: Specifies the list of pins or ports associated with an ideal clock waveform of the clock. If you do not specify this option, you must specify the `-name` *`clock_name`* option, which creates a virtual clock not associated with a port or pin. If you specify a clock on a pin that already has a clock, the new clock replaces the old one.

`-waveform` *`edge_list`*
: Specifies the rise and fall edge times of the clock waveforms in library time units over an entire clock period. The first value in the edge list is the time at which the first transition on the clock occurs, which is the first rising transition after time zero, and edges are monotonically increasing. The numbers should represent one full clock period. If you do not specify the `-waveform` *`edge_list`* option, a default waveform is assumed, and the leading edge is placed at `0` and the trailing edge is placed at the midpoint of the period, such that a symmetric clock is generated.

Examples

- The following command creates an `ideal_clk1` clock on port `clkA` with a period of `5`, a rise at `0`, and a fall at `2.5`:

  ```
  create_clock [get_ports {clkA}] -name ideal_clk1 -period 5 -waveform {0 2.5}
  ```

- The following command creates an `ideal_clk2` clock on port `clkB` with a period of `10`, a rise at `1`, and a fall at `9`:

  ```
  create_clock [get_ports {clkB}] -name ideal_clk2 -period 10 -waveform {1 9}
  ```

Related Information

`all clocks`

create_generated_clock

```
create_generated_clock
     [-name clock_signame]
     -source source_pin
     {-multiply_by integer | -divide_by integer | -edges edge_list
       | -combinational}
     [-edge_shift edge_shift_list]
     [-duty_cycle percent]
     target_pin_list
     [-master_clock source_clock_name]
     [-add]
     [-invert]
```

Creates a new clock signal from the clock waveform of a given pin in the design, and binds it with the pins or hierarchical pins in the *target_pin_list* argument. Whenever the source clock changes, the derived clock(s) change automatically.

Generate the new clock waveform using one of three ways:

- Multiply the frequency of the source clock
- Divide the frequency of the source clock
- Select the edges of the source clock to be mapped to the edges of the new clock

Note: A `create_generated_clock` assertion on a pin overrides any existing `create_generated_clock` assertion if both the constraints have the same source pin. In the following command sequence, the second assertion overrides the first:

```
create_generated_clock -name GCK1 -source CK -divide_by 2 U_FD1/Q
create_generated_clock -name GCK2 -source CK -multiply_by 2 U_FD1/Q
```

To have both assertions applied, you must specify the `-add` parameter. For example:

```
create_generated_clock -name GCK1 -source CK -divide_by 2 U_FD1/Q
create_generated_clock -name GCK2 -add -master_clock CLK -source CK
-multiply_by 2    U_FD1/Q
```

Parameters

`-add`	Models multiple generated clocks on the same source when multiple clocks must fan into the source pin. Ideally, one generated clock must be specified for each clock that fans into the master pin. Specify this option with the `-name` and `-master_clock` options. By default, the software creates one generated clock at the pin by using the fastest clock present on the source pin as the master clock. However, use the `-add` option to specify a different clock name for each generated clock when used with the -`master_clock` option. Subsequently, you can use this clock name for setting other constraints, such as the `set_false_path` command and the `set_input_delay` command.
`-combinational`	Restricts the path search for latency to only paths which do not cross sequential devices, or the source pins of other generated clocks. When you specify `-combinational`, if there are both sequential and combinational latency paths, the software only considers the combinational paths for the generated clock. If there are no combinational latency paths for the generated clock, the generated clock does not inherit latency from its parent clock hierarchy. By default, when calculating the latency of a generated clock, the software searches for the longest (and shortest) paths from the parent waveform to the generated clock point. All paths are considered, including those which traverse sequential elements such as flip-flops, registers, and latches. The default behavior is intended for cases where there is a cascade of generated clocks, and some of the clocks are not explicitly defined. The `-combinational` option can be used in conjunction with the `-multiply_by`, `-divide_by`, or `-edges` options by setting the `timing_enable_genclk_edge_based_source_latency` global variable to `true`. By default, the variable is set to `false`.
`-divide_by` *`integer`*	Determines the frequency of the new clock by dividing the frequency of the source clock by *`dfactor`*. For example, if *`dfactor=5`*, the new frequency is 1/5 of the source frequency, and the new clock period is 5 times the source clock period.

`-duty_cycle` *`percent`*

Sets the duty cycle (high pulse width or clock period). The number (*`percent`*) is between `0` and `100`. For example, if *`percent=50`*, the high pulse width of the new clock waveform is the same as its low pulse width.

Note: The `-duty_cycle` option can only be used with the `-multiply_by` option, otherwise it will be ignored with a warning.

`-edges` *`edge_list`* Selects a list of edges from the source clock that form the edges of the derived clock. Currently, three edge numbers are allowed. See the frequency division example below.

`-edge_shift` *`edge_shift_list`*

Specifies the amount of shift for each edge in the *`edge_list`* option.

`-invert` Inverts the source clock. Use with the `-source` option and the `-multiply_by` or `-divide_by` options.

`-master_clock` *`source_clock_name`*

Generates a target clock from the specified clock name. If you specify the `-master_clock` option and multiple signals arrive at the source pin, the clock specified by this option is used to generate the target clock. Use the `-add` parameter along with `-master_clock` parameter to add a clock to the existing clock when you want to specify multiple clocks on the same source for simultaneous analysis.

`-multiply_by` *`integer`*

Determines the frequency of the new clock by multiplying the frequency of the source clock by *`mfactor`*. For example, if *`mfactor=4`*, the new frequency is 4 times the source frequency, and the new clock period is 1/4 of the source clock period.

`-name` *`clock_signame`*

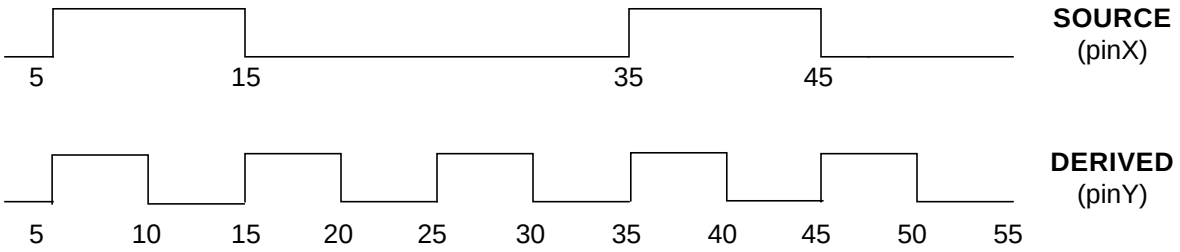
	Specifies the name for the generated clock. *Default*: The system creates a name for you. Always specify a name for the generated clock. Some commands issue an error if you use the system-generated name.
`-source` *`source_pin`*	Specifies the name of the source pin from which the new clock is to be derived. This is a mandatory option.
`target_pin_list`	Specifies the list of pins where the generated clock assertion is applied. The clock signal propagating downstream from each of these pins becomes associated with the generated clock.

Examples

- The following command generates the derived clock waveform, as shown in Figure 30-1:

```
create_generated_clock -name mult_3 -source pinX -multiply_by 3 -duty_cycle 50
pinY
```

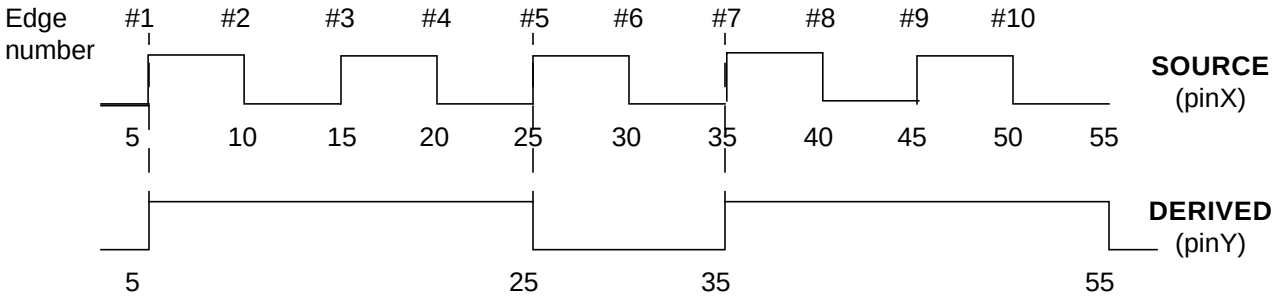
Figure 30-1 Derived Clock (Frequency Multiplication)



- The following command shows one way to create a divide-by-3 clock by using edge numbers of the source waveform to specify the new waveform, as shown in Figure 30-2,:

```
create_generated_clock -name genclk -source pinX -edges {1 5 7} pinY
```

Figure 30-2 Derived Clock (Frequency Division)

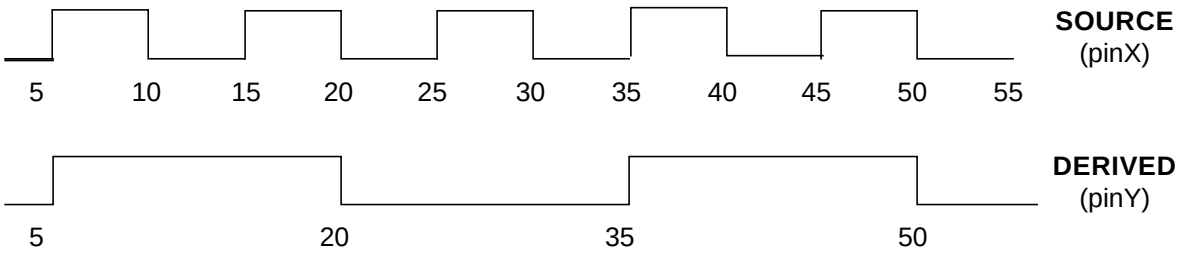


- The following command uses the `-divide_by` option to get the same clock period; the difference is in the duty cycle:

  ```
  create_generated_clock -name div_3 -source pinX -divide_by 3 pinY
  ```

  With the `-edges` option, the above example creates a clock with a duty cycle of 67 percent (20/30). However, using the `-divide_by` 3 option preserves the 50 percent duty cycle of the source clock as shown in Figure 30-3.

Figure 30-3 Derived Clock (Frequency Division, 50 Percent Duty-cycle)



- Using the following commands, when there is more than 1 source clock, the tool creates the generated clock based on the source clock specified with the `-master` option:

  ```
  create_clock c1 -p 10 [get_ports {ck} ] -add
  create_clock c2 -p 10 [get_ports {ck} ] -add
  create_generated_clock -name newclk -source ck -divide_by 2 -master_clock c1
  reg/Q -add
  ```

  Using the `report_clocks` command, the output waveform of the `newclk` clock is a divide-by-2 of the `c1` clock and is reported as shown in Example 30-1.

Example 30-1 report_clocks after using the -master option

```
+-------------------------------------+
|         Clock Descriptions          |
|-------------------------------------|
|  Clock |  Period |  Lead  |  Trail  |
|  Name  |         |        |         |
|--------+---------+--------+---------|
|  c1    | 10.0000 | 0.0000 |  5.0000 |
|  c2    | 20.0000 | 0.0000 | 10.0000 |
| newclk | 20.0000 | 0.0000 | 10.0000 |
+-------------------------------------+
```

Related Information

create_clock

get_clocks

report_clocks

current_design

```
current_design
     [none | design_name]
```

Sets or returns the current design.

Parameter

design_name	Specifies the name of the top design. **Note:** Setting current design to a sub-design is not supported.

Examples

- The following command returns the name of the current design:

  ```
  current_design
  ```

current_instance

```
current_instance
     [hierarchical_instance]
```

Sets the instance given by *hierarchical_instance* to be the current instance. Design objects referenced by subsequent commands can be found relative to this instance. All searches for design objects are started in this instance. If design objects are referenced hierarchically, *hierarchical_instance* is used as the root (top) of the hierarchy.

Parameters

hierarchical_instance

Specifies the name of the instance to become the current instance. If *hierarchical_instance* is not given, the current instance is set to the top instance of the hierarchy.

Examples

- The following command sets the current instance to MAC/MULTI, sets a constant on a register pin inside the hierarchical instance and then later sets the current instance back to top:

```
current_instance MAC/MULT1
set_case_analysis 0 [get_pins {reg6/SE}]
...
current_instance
```

group_path

```
group_path
     -name path_group_name
     [-from from_list | -to to_list | -through through_list]
```

Groups paths in a design, and identifies them with a path group name. You can use the path group names for reporting.

Parameters

`-from` *`from_list`*
: Specifies a list of pins, instances, or clocks that are the start of the paths. When you specify a clock object, all paths that are triggered by the clock are included in the path group.

`-name` *`path_group_name`*
: Specifies the name of the path group. If the path_group_name already exists, the specified paths are added to the existing path group.

`-through` *`through_list`*
: Specifies the pins or instances where the paths go through. When you specify multiple through list, the paths that go through the objects in the specified order are added to the path group.

`-to` *`to_list`*
: Specifies a list of pins, instances, or clocks where the path ends. You must specify valid endpoints in the *`to_list`* or else the software ignores the constraint. When you specify a clock object, all paths that are checked by the clock are included in the path group. If you specify an instance name in the *`to_list`*, all the data input pins of the instance are added to the path group.

Example

- The following example groups paths that start from specific pins and end at the D pin of any instances whose name start with the string ff. The example also reports the specified path groups:

  ```
  group_path -name path1 -from {ff1/CLK ff2/CLK} -to {ff*/D}
  report_timing -path_group path1
  ```

- The following command groups paths that go through specific points.

```
group_path -name path2 -through an1/Z -through an2/Z -through an3/Z
```

reset_annotated_check

```
reset_annotated_check
     [-clock [rise | fall]]
     [-setup | -recovery]
     [-hold | -removal]
     [-nochange_high | -nochange_low]
     [-rise] [-fall]
     [[-from from_pins][-to to_pins]] | -all
     object_list
```

Removes annotated timing check values that you had set by using the `set_annotated_check` command or by SDF annotation.

Note: The reset_annotated_check command does not reset the timing check values set by SDF annotations.

Parameters

`-all`	Resets all annotated timing check values in the design.
`-clock [rise | fall]`	
	Specifies the direction of the related pin of the check to reset.
	Default: Resets checks against both edges of the clock
`-fall`	Restricts the reset to only affect the falling edge of the constrained pins.
	Default: Checks on both rising and falling signals are reset
`-from` *`from_pins`*	Specifies the beginning instance pin of the timing arcs for which the annotated check is reset. This is usually the reference (for example, clock pin) of the constraint. The selected timing arcs are those that have their source pins specified using this option and the sink pins specified using the `-to` option. If this option is not specified, only timing arcs whose sink pins are specified using the `-to` option are selected. You cannot use this parameter with *`object_list`* parameter.
`-hold | -removal`	Resets annotations on the hold or removal check type.
`-nochange_high | -nochange_low`	
	Resets the annotations on the specified timing check type.
`object_list`	Resets the annotated checks on specified list or collection of cell instances.

`-rise`	Restricts the reset to only affect the rising edge of the constrained pins. *Default:* Checks on both rising and falling signals are reset
`-setup | -recovery`	Resets annotations on the setup or recovery check type.
`-to` *`to_pins`*	Specifies the terminating instance pin of the timing arcs for which the annotated check is reset. This is usually the constrained pin (data pin) of the check. The selected timing arcs are those that have their sink pins specified using this option and the source pins specified using the `-from` option. If this option is not specified, only timing arcs whose source pins are specified using the `-from` option are selected. You cannot use this parameter with *`object_list`* parameter.

Examples

- The following command resets all the timing check annotations in the design:

  ```
  reset_annotated_check -all
  ```

- The following command resets all timing check annotations on a particular cell:

  ```
  reset_annotated_check [get_cells BLK/BR1]
  ```

- The following command resets specific checks on a specific instance:

  ```
  reset_annotated_check -setup -hold [get_cells BLK/BR2]
  ```

- The following command resets checks on a specific timing check arc:

  ```
  reset_annotated_check -from BLK/BR1/CK -to BLK/BR1/D
  ```

reset_case_analysis

```
reset_case_analysis
     list_of_ports_or_pins
```

Removes the assertions set by the set_case_analysis command.

Parameters

list_of_ports_or_pins

Specifies the list of ports or pins for which the assertions are to be removed.

Examples

- The following commands sets the assertion to 0 and then resets them:

```
set_case_analysis 0 CM1/S0
get_property [get_pins CM1/S0] constant_value
```

Returns a value of 0

```
reset_case_analysis [get_pins CM1/S0]
get_property [get_pins CM1/S0] constant_value
```

Returns a value of NA

reset_clock

```
reset_clock
     [-all | clock_list]
```

Removes previously created clock assertions.

Parameters

`-all`	Removes all clocks in the design.
`clock_list`	Specifies the list of clocks for which the clock assertion is removed.

Example

- The following commands resets a list of clocks and all clocks, and display the results using the `report_clocks` command:

```
report_clocks
```

Generates the following report:

```
+-------------------------------------------------------------------+
|                        Clock Descriptions                         |
|-------------------------------------------------------------------|
|       |        |        |       |       |       Attributes       |
|-------+--------+--------+-------+-------+------------------------|
| Clock | Source | Period |  Lead | Trail | Generated | Propagated |
|  Name |        |        |       |       |           |            |
|-------+--------+--------+-------+-------+-----------+------------|
|  CCLK |        | 10.000 | 0.000 | 5.000 |     n     |     n      |
|  VCLK |        | 10.000 | 0.000 | 5.000 |     n     |     n      |
|  WAVE |  tclk  | 10.000 | 0.000 | 5.000 |     n     |     n      |
+-------------------------------------------------------------------+
```

```
reset_clock [get_clocks CCLK]
report_clocks
```

Generates the following report:

```
+-------------------------------------------------------------------+
|                        Clock Descriptions                         |
|-------------------------------------------------------------------|
|       |        |        |       |       |       Attributes       |
|-------+--------+--------+-------+-------+------------------------|
| Clock | Source | Period |  Lead | Trail | Generated | Propagated |
|  Name |        |        |       |       |           |            |
|-------+--------+--------+-------+-------+-----------+------------|
|  VCLK |        | 10.000 | 0.000 | 5.000 |     n     |     n      |
|  WAVE |  tclk  | 10.000 | 0.000 | 5.000 |     n     |     n      |
+-------------------------------------------------------------------+
```

```
reset_clock -all
report_clocks

# This does not return anything indicating that all clocks were reset.
```

Related Information

- report_clocks

reset_clock_latency

```
reset_clock_latency
     [-source]
     object_list
```

Removes the assertions that were made by previous set_clock_latency commands.

Parameters

object_list	Specifies a list of pins, ports or clock objects for which the assertions are removed.
-source	Removes the source latency.

Examples

- The following commands show the use of the reset_clock_latency command to remove a leaf-level set_clock_latency assertion, after the clock tree has been put into propagated mode:

```
set_clock_latency 3.141 [get_pins BLK/BR2/CK]
report_timing -to BLK/BR2/D -path_type full_clock
```

The following report is generated:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint:   BLK/BR2/D (v) checked with  leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by  leading edge of 'WAVE'
Other End Arrival Time          3.141 <---
- Setup                         0.191
+ Phase Shift                  10.000
= Required Time                12.950
- Arrival Time                  0.249
= Slack Time                   12.701
...
```

```
set_propagated_clock [get_ports tclk]
report_timing -path_type full_clock -net -to BLK/BR2/D
```

The following report is generated:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint:   BLK/BR2/D (v) checked with  leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by  leading edge of 'WAVE'
Other End Arrival Time          3.141 <--------
- Setup                         0.191
+ Phase Shift                  10.000
= Required Time                12.950
- Arrival Time                  0.594
= Slack Time                   12.356
...
```

In this example, setting the root of the clock tree to propagated mode is not sufficient to override the `set_clock_latency` assertion at the clock endpoint. You must use the `reset_clock_latency` command to remove the assertion, so that the register sees propagated clock timing.

```
reset_clock_latency [get_pins BLK/BR2/CK]
report_timing -path_type full_clock -net -to BLK/BR2/D
```

The following report is generated:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint:   BLK/BR2/D (v) checked with  leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by  leading edge of 'WAVE'
Other End Arrival Time          0.345 <--------
- Setup                         0.186
+ Phase Shift                  10.000
= Required Time                10.159
- Arrival Time                  0.588
= Slack Time                    9.571
```

- The following commands show that `reset_clock_latency` is only effective when it is applied on the same object for which you had set the assertion using the `set_clock_latency` command:

Latency here is specified on a pin object

```
set_clock_latency 3.141 [get_pins BLK/BR2/CK]
report_timing -path_type full -net -to BLK/BR2/D
```

The following report is generated:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint:   BLK/BR2/D (v) checked with  leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by  leading edge of 'WAVE'
Other End Arrival Time          3.141 <-
- Setup                         0.191
+ Phase Shift                  10.000
= Required Time                12.950
- Arrival Time                  0.249
= Slack Time                   12.701
```

Resetting of the latency is done on a clock object and has no effect on downstream latencies specified on pin objects

In this example, a reset_clock_latency on the clock waveform that is driving BLK/BR2/CK will have no effect.

```
reset_clock_latency [get_clocks WAVE]
report_timing -path_type full -net -to BLK/BR2/D
```

The following report is generated:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint:   BLK/BR2/D (v) checked with  leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by  leading edge of 'WAVE'
Other End Arrival Time          3.141  <-
- Setup                         0.191
```

```
+ Phase Shift                10.000
= Required Time              12.950
- Arrival Time                0.249
= Slack Time                 12.701
```

reset_clock_transition

```
reset_clock_transition
     clock_list
```

Removes the clock transition assertions you had set on the specified list of clock waveforms.

Parameters

clock_list	Specifies a list of clock waveforms for which the clock transition assertions are removed. The *clock_list* variable can be a Tcl list or collection of clock waveforms.

Examples

- The following commands set the clock transition assertions, and resets them and displays the results using the `report_timing` command:

```
set_clock_transition 0.123 [get_clocks WAVE]
report_timing -net -to BLK/BR2/D
```

Generates the following report

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint:   BLK/BR2/D (v) checked with  leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by  leading edge of 'WAVE'
Other End Arrival Time          0.000
- Setup                         0.192
+ Phase Shift                  10.000
= Required Time                 9.808
- Arrival Time                  0.249
= Slack Time                    9.559
Clock Rise Edge                      0.000
= Beginpoint Arrival Time            0.000
+-------------------------------------------------------------------------+
|  Instance  |  Cell   |     Arc     |  Pin  |  Slew | Delay | Arrival |
|            |         |             |       |       |       |  Time   |
|------------+---------+-------------+-------+-------+-------+---------|
|            |         | tclk ^      | tclk  | 0.123 |       |   0.000 |
| CG/BC1     | BUFX2   |             | A     | 0.123 | 0.000 |   0.000 |
| CG/BC1     | BUFX2   | A ^ -> Y ^  | Y     | 0.123 | 0.000 |   0.000 |
| TC1        | BUFX2   |             | A     | 0.123 | 0.000 |   0.000 |
| TC1        | BUFX2   | A ^ -> Y ^  | Y     | 0.123 | 0.000 |   0.000 |
| CG1        | AND2X4  |             | B     | 0.123 | 0.000 |   0.000 |
| CG1        | AND2X4  | B ^ -> Y ^  | Y     | 0.123 | 0.000 |   0.000 |
| BLK/blkint | BUFX2   |             | A     | 0.123 | 0.000 |   0.000 |
| BLK/blkint | BUFX2   | A ^ -> Y ^  | Y     | 0.123 | 0.000 |   0.000 |
| BLK/BR1    | DFFHQX1 |             | CK    | 0.123 | 0.000 |   0.000 |
```

```
reset_clock_transition [get_clocks WAVE]
report_timing -net -to BLK/BR2/D
```

Generates the following report:

```
Path 1: MET Setup Check with Pin BLK/BR2/CK
Endpoint:   BLK/BR2/D (v) checked with  leading edge of 'WAVE'
Beginpoint: BLK/BR1/Q (v) triggered by  leading edge of 'WAVE'
Other End Arrival Time          0.000
- Setup                         0.191
+ Phase Shift                  10.000
= Required Time                 9.809
- Arrival Time                  0.249
= Slack Time                    9.560
Clock Rise Edge                      0.000
= Beginpoint Arrival Time            0.000
+-------------------------------------------------------------------------+
|  Instance  |  Cell   |     Arc     |  Pin  |  Slew | Delay | Arrival |
|            |         |             |       |       |       |  Time   |
|------------+---------+-------------+-------+-------+-------+---------|
|            |         | tclk ^      | tclk  | 0.000 |       |   0.000 |
| CG/BC1     | BUFX2   |             | A     | 0.000 | 0.000 |   0.000 |
| CG/BC1     | BUFX2   | A ^ -> Y ^  | Y     | 0.000 | 0.000 |   0.000 |
| TC1        | BUFX2   |             | A     | 0.000 | 0.000 |   0.000 |
| TC1        | BUFX2   | A ^ -> Y ^  | Y     | 0.000 | 0.000 |   0.000 |
| CG1        | AND2X4  |             | B     | 0.000 | 0.000 |   0.000 |
| CG1        | AND2X4  | B ^ -> Y ^  | Y     | 0.000 | 0.000 |   0.000 |
| BLK/blkint | BUFX2   |             | A     | 0.000 | 0.000 |   0.000 |
| BLK/blkint | BUFX2   | A ^ -> Y ^  | Y     | 0.000 | 0.000 |   0.000 |
| BLK/BR1    | DFFHQX1 |             | CK    | 0.000 | 0.000 |   0.000 |
```

reset_clock_tree_latency

```
reset_clock_tree_latency
     pin_port_list | clock_list
```

Resets all network clock latency set using the set_clock_latency command in the fanout of the specified clock or pins and ports. The reset_clock_tree_latency command resets the latency until the root pins of clocks and generated clocks. Use this command after building clock tree and setting the clock network from the root to propagated mode using the `set_propagated_clock` *`clockRootPinOrPort`* command.

Note: You can use the reset_clock_tree_latency command only on command line and command script. You cannot specify it in an SDC file.

Parameters

`clock_list`	Resets latencies downstream for all associated clock root ports or points.
`pin_port_list`	Resets latency assertions downstream from the specified point.

reset_clock_uncertainty

```
reset_clock_uncertainty
     [-setup | -hold]
     {  {-from | -rise_from | -fall_from} clksig_from_list
        {-to | -rise_to | -fall_to} clksig_to_list
      | pin_or_clock_list
     }
```

Removes the assertions that were made by previous `set_clock_uncertainty` commands.

Parameters

`-from | -rise_from | -fall_from` *`clksig_from_list`*

Removes the assertions for uncertainty value for the specified list of launching clocks.

Note: You must specify one of these parameters with one of the `-to`, `-rise_to`, or `-fall_to` parameters. You cannot use these parameters when *`pin_or_clock_list`* is specified.

`-fall_from` Removes the uncertainty value only from the falling edge of the launching clocks.

`-from` Removes the uncertainty value from both the rising and falling edges of the launching clocks.

`-rise_from` removes the uncertainty value only from the rising edge of the launching clocks.

pin_or_clock_list Removes the target uncertainty assertions from specified list of clocks or pins. The uncertainty value is removed from the registers clocked by the target waveform, or register pins in the fanout of the specified pin.

You can specify a list of clock names, or clock root pin or port names relative to the current module. You also can specify a collection of clocks, or clock root pins or ports.

Specifying a register or latch clock pin, or an arbitrary pin in the clock tree for *pin_or_clock_list* currently is not supported.

Note: You cannot use this parameter when any of the `-from` and `-to` parameter pairs are specified.

`-setup | -hold` Specifies whether the clock uncertainty is removed for setup or hold checks.

Default: Removes clock uncertainty for both setup and hold check.

`-to | -rise_to | -fall_to` *clksig_to_list*

Removes the uncertainty value from the specified list of capture clocks.

Note: You must specify one of these parameters with one of the `-from`, `-rise_from`, or `-fall_from` parameters. You cannot use these parameters if *pin_or_clock_list* is specified.

`-fall_to` Removes the uncertainty value only from the falling edge of the capture clocks.

`-rise_to` Removes the uncertainty value only from the rising edge of the capture clocks.

`-to` Removes the uncertainty value from both the rising and falling edges of the capture clocks.

Example

- The following commands sets and resets the clock uncertainty assertions and displays the results using the `report_timing` command:

```
set_clock_uncertainty 0.345 -from [get_clocks VCLK] -to [get_clocks WAVE]
report_timing -net -from tin
```

Generates the following report:

```
Path 1: MET Setup Check with Pin TR1/BR1/CK
Endpoint:   TR1/BR1/D (v) checked with  leading edge of 'WAVE'
Beginpoint: tin       (v) triggered by trailing edge of 'VCLK'
Other End Arrival Time          0.000
- Setup                         0.192
+ Phase Shift                  10.000
- Uncertainty                   0.345  <---
= Required Time                 9.463
- Arrival Time                  5.877
= Slack Time                    3.586


reset_clock_uncertainty -from [get_clocks VCLK] -to [get_clocks WAVE]
report_timing -net -from tin
```

Generates the following report:

```
Path 1: MET Setup Check with Pin TR1/BR1/CK
Endpoint:   TR1/BR1/D (v) checked with  leading edge of 'WAVE'
Beginpoint: tin       (v) triggered by trailing edge of 'VCLK'
Other End Arrival Time          0.000
- Setup                         0.192
+ Phase Shift                  10.000
= Required Time                 9.808
- Arrival Time                  5.877
= Slack Time                    3.931
```

Related Information

set_clock_uncertainty

reset_data_check

```
reset_data_check
     [{-from | -rise_from | -fall_from} pin_or_port_list]
     [{-to | -rise_to | -fall_to} pin_or_port_list]
     [-setup | -hold]
     [-clock clock_object]
```

Removes specified data-to-data checks that you have specified using the `set_data_check` command.

Parameters

`-clock` *`clock_object`*

Removes the data check with a related (`-from`) pin event that is triggered by the specified clock.

Use this parameter only if the original `set_data_check` constraint also used the `-clock` parameter to specify an explicit clock relationship.

`-from |-rise_from |` `-fall_from` *`pin_or_port_list`*

Specifies the related pins of the data-to-data checks to be removed.

The *`pin_or_port_list`* list can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified.

Use only one `-from`, `-rise_from`, or `-fall_from` parameter per command.

By default, the `-from` parameter removes checks for both the rising and the falling edges.

Using `-rise_from` parameter removes checks at the rising edge of the signal on the from pins.

Using `-fall_from` parameter removes checks at the falling edge of the signal on the from pins.

`-setup | -hold`

Specifies whether the data check to be removed is for setup or hold check. If you do not specify either `-setup` or `-hold` option, both setup and hold checks are removed.

`-to | -rise_to | -fall_to` *`pin_or_port_list`*

Specifies the constrained pins of data-to-data checks to be removed.

The *`pin_or_port_list`* list can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified.

Use only one `-to`, `-rise_to`, or `-fall_to` parameter per command.

By default, the `-to` parameters removes both the rising and the falling edges.

Using `-rise_to` parameter removes the rising edge of the signal on the to pins.

Using `-fall_to` parameter removes the falling edge of the signal on the to pins.

Examples

- The following commands set and reset the data checks from `port30` to `port28`:

```
set_data_check  0.345 -rise_from port30 -to port28
report_timing  -rise_from port30 -to port28
```

The following report is displayed:

```
Path 1: VIOLATED Data To Data Setup Check with Pin port30
Endpoint:   port28 (v) checked with  leading edge of 'PH1'
Beginpoint: U2/Q   (v) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.465
- Data Check Setup              0.345
+ Phase Shift                   0.000
= Required Time                 0.120
- Arrival Time                  0.309
= Slack Time                   -0.189
```

```
reset_data_check -rise_from port30 -to port28
report_timing  -rise_from port30 -to port28
```

The following information is displayed:

```
No constrained timing paths with given description found.
Paths may be unconstrained (try '-unconstrained' option) or may not exist.
```

reset_disable_timing

```
reset_disable_timing
     [-from pin_name -to pin_name]
     object_list
```

Restores timing arcs that were disabled using the set_disable_timing command.

Parameters

`-from` *`pin_name`* `-to` *`pin_name`*

Specifies the arc that is to be restored. Specify the `-from` and `-to` options together.

You can specify the `-from` and `-to` parameters when the *`object_list`* contains instance or library cell objects. Do not specify the `-from` and `-to` parameters when the *`object_list`* includes instance pin or library cell pin objects.

The `-from` and `-to` options are optional. If not specified, all cell arcs found in the instance or library cell objects are restored.

`object_list`

Specifies the instance, instance pin, library cell, or library cell pin objects.

If the *`object_list`* includes instance pin or library cell pin objects, all cell arcs attached to the specified pin objects are restored.

If the *`object_list`* contains instance or library cell objects, and the `-from` and `-to` parameters are not specified, all arcs found in the specified instance or library cell objects are restored.

Examples

- The following commands sets and resets the timing arcs and displays the resulting reports:

```
set_disable_timing -from A -to Y [get_cells U0]
report_inactive_arcs
```

The following report is displayed:

```
+-------------------------------------------------------------------+
|Flags:|const              |propagated constant     |
|      |snipped            |loop snipped arcs       |
```

```
|      |disable              |set_disable_timing      |
|      |disable_clock_gating|set_disable_clock_gating|
|      |library_disable     |set_disable_cell_timing |
|      |Missing_Phase       |No Arc Phase Data       |
|From  | To                 | DisableType            |ArcType  |  Reason    |
|------+--------------------+------------------------+---------+------------|
|U0/A ^| U0/Y ^             | disable            |combinational|User Disable|
|U0/A v| U0/Y v             | disable            |combinational|User Disable|
+-------------------------------------------------------------------------------+


reset_disable_timing -from A -to Y [get_cells U0]
report_inactive_arcs
```

The following report is displayed:

```
+-------------------------------------------------------------------------+
|Flags:| const               | propagated constant      |
|      | snipped             | loop snipped arcs        |
|      | disable             | set_disable_timing       |
|      | disable_clock_gating| set_disable_clock_gating |
|      | library_disable     | set_disable_cell_timing  |
|      | Missing_Phase       | No Arc Phase Data        |
|From  | To                  | DisableType              | ArcType | Reason |
|------+---------------------+--------------------------+---------+--------|
```

- The following commands disables the timing arcs from `CK` to `Q` for instance `U2`, then resets the assertions using the instance reference:

```
set_disable_timing -from CK -to Q [get_cells U2]
report_inactive_arcs
```

The following report is displayed:

```
+-------------------------------------------------------------------------------+
|Flags: |const               |propagated constant     |
|       |snipped             |loop snipped arcs       |
|       |disable             |set_disable_timing      |
|       |disable_clock_gating|set_disable_clock_gating|
|       |library_disable     |set_disable_cell_timing |
|       |Missing_Phase       |No Arc Phase Data       |
|From   |To                  |DisableType             |ArcType  |Reason      |
|-------+--------------------+------------------------+---------+------------|
|U2/CK ^|U2/Q ^              | disable            |rising_edge|User Disable|
|U2/CK ^|U2/Q v              | disable            |rising_edge|User Disable|
+-------------------------------------------------------------------------------+

reset_disable_timing U2
report_inactive_arcs
```

The following report is displayed:

```
+-------------------------------------------------------------------------------+
|Flags: |const               |propagated constant     |
|       |snipped             |loop snipped arcs       |
|       |disable             |set_disable_timing      |
|       |disable_clock_gating|set_disable_clock_gating|
|       |library_disable     |set_disable_cell_timing |
|       |Missing_Phase       |No Arc Phase Data       |
```

From	To	DisableType	ArcType	Reason

reset_generated_clock

```
reset_generated_clock
     [-all]
     target_clock_list
```

Removes generated clock assertions previously created using the `create_generated_clock` command.

Parameters

`-all`	Removes all generated clocks in the design.
`target_clock_list`	Specifies the list of clocks where the generated clock assertion is removed.

Example

- The following command removes the generated clock assertions on all generated clocks `RC/Q`:

  ```
  report_clocks -generated
  ```

  # The following report is displayed:

  ```
  +-------------------------------------------------------------------+
  |                         Clock Descriptions                        |
  |-------------------------------------------------------------------|
  |       |        |        |       |        |       Attributes       |
  |-------+--------+--------+-------+--------+------------------------|
  | Clock | Source | Period |  Lead |  Trail | Generated | Propagated |
  |  Name |        |        |       |        |           |            |
  |-------+--------+--------+-------+--------+-----------+------------|
  |  PH1  |  clk   | 10.000 | 0.000 |  5.000 |     n     |     y      |
  |  RC/Q |  RC/Q  | 20.000 | 0.000 | 10.000 |     y     |     y      |
  +-------------------------------------------------------------------+

  +-------------------------------------------------------------------------+
  |                      Generated-Clock Descriptions                       |
  |-------------------------------------------------------------------------|
  |Name|Generated|Master|Master-clock|Inverted|Freq|Duty-Cycle|Edges|Edge-Shift
  |    |Source(pin)|Source(pin)|     |  |Multiplier|          |     |        |
  |----+---------+------+------------+--------+----+----------+-----+--------|
  |RC/Q|RC/Q     |clk   |   PH1      |n    | 1/2  |     -      |   -   | -     |
  +-------------------------------------------------------------------------+
  ```

  # Note, in this case the name of the generated clock is RC/Q as the -name option was omitted when creating the clock

```
reset_generated_clock  [get_clocks RC/Q]
report_clocks -generated
```

The following report is generated:

```
+-------------------------------------------------------------------+
|                         Clock Descriptions                        |
|-------------------------------------------------------------------|
|       |        |        |       |       |       Attributes       |
|-------+--------+--------+-------+-------+------------------------|
| Clock | Source | Period |  Lead | Trail | Generated | Propagated |
|  Name |        |        |       |       |           |            |
|-------+--------+--------+-------+-------+-----------+------------|
|  PH1  |  clk   | 10.000 | 0.000 | 5.000 |     n     |     y      |
+-------------------------------------------------------------------+
```

Related Information

create_generated_clock

reset_input_delay

```
reset_input_delay
     [-clock clk_name]
     [-clock_fall]
     [-rise] [-fall]
     [-max] [-min]
     pin_or_port_list
```

Resets previously specified input delay assertions.

Parameters

`-clock` *`clock_name`*	Specifies the name of the clock. *Default*: The asynchronous (@) clock.
`-clock_fall`	Removes the assertions for the delay that is relative to the falling edge of the clock. You must specify the `-clock` parameter with this parameter. *Default:* Removes the delay assertions relative to the rising edge.
`-fall`	Resets the input delay for the falling edge at the input port. If both `-rise` and `-fall` options are omitted, the input delay is reset on both the edges.
`-max`	Removes the delay assertions that refer to the setup analysis. If you do not specify the `-max` parameter, the input delay assertions for both setup and hold analysis are removed.
`-min`	Removes the delay assertions that refer to the hold analysis. If you do not specify the `-min` parameter, the input delay assertions for both setup and hold analysis are removed.
`pin_or_port_list`	Specifies a single or multiple pins or ports for which the input delay is reset. To specify multiple pins, enclose the list with curly braces (`{}`) and separate the pin information with white space.
`-rise`	Resets the input delay for the rising edge at the input port. If both `-rise` and `-fall` options are omitted, the input delay is reset on both the edges.

Related Information

`set_input_delay`

reset_mode

```
reset_mode
     [-type cell]
     list_of_modes
     instance_list
```

Resets the Liberty timing library modes for a specified instance.

Library modes are set using the set_mode command. By default, all library modes in the timing library are active. If you specify one or more library modes using the set_mode command, only the timing arcs consistent with the specified mode(s) are active for the instance. All other library modes become inactive. When you reset the library mode for the instance, all library modes become active again.

Parameters

instance_list	Specifies the instances to which to reset the library modes.
list_of_modes	Specifies the name of the library modes to reset.
-type cell	Resets the library mode at the instance level. **Note:** Setting the library mode at the instance level is the only form of set_mode that is currently supported. This parameter exists to support constraints coming from external sources.

Examples

- The following command specifies that only the timing arcs consistent with the tst_mode_disable library mode are active for instances of i_ram. The tst_mode_enable library mode is made inactive.

  ```
  set_mode tst_mode_disable [get_cells i_ram]
  ```

- The following command reports the status of all library modes associated with i_ram. It shows that tst_mode_disable is active, and tst_mode_enable is inactive.

  ```
  report_mode i_ram
  ```

```
+-------------------------------------------------+
|         Mode Group testmode of i_ram            |
|-------------------------------------------------|
|    Mode Name      |  Status   |  Condition      |
|-------------------+-----------+-----------------|
|  tst_mode_enable  | Inactive  |  TST_BYPASS     |
| tst_mode_disable  | ACTIVE    |  !(TST_BYPASS)  |
+-------------------------------------------------+
```

- The following command resets the library mode for instances of U1:

  ```
  reset_mode tst_mode_disable i_ram
  ```

  If you generate a report of the library mode status for `i_ram`, it shows that both modes are now active:

  ```
  report_mode i_ram
  ```

  ```
  +-------------------------------------------------+
  |         Mode Group testmode of i_ram            |
  |-------------------------------------------------|
  |    Mode Name      |  Status   |  Condition      |
  |-------------------+-----------+-----------------|
  |  tst_mode_enable  | ACTIVE    |  TST_BYPASS     |
  | tst_mode_disable  | ACTIVE    |  !(TST_BYPASS)  |
  +-------------------------------------------------+
  ```

reset_output_delay

```
reset_output_delay
     [-clock clock_name]
     [-clock_fall]
     [-rise] [-fall]
     [-max] [-min]
     pin_or_port_list
```

Resets previously specified output delay assertions.

Parameters

`-clock` *`clock_name`*	Specifies the name of the clock. *Default*: The asynchronous (@) clock.
`-clock_fall`	Removes the assertions for the delay that is relative to the falling edge of the clock. You must specify the `-clock` parameter with this parameter. *Default:* Removes the delay assertions relative to the rising edge.
`-fall`	Resets the output delay for the falling edge at the output port. If both `-rise` and `-fall` options are omitted, the output delay is reset on both the edges.
`-max`	Removes the delay assertions that refer to the setup analysis. If you do not specify the `-max` parameter, the output delay assertions for both setup and hold analysis are removed.
`-min`	Removes the delay assertions that refer to the hold analysis. If you do not specify the `-min` parameter, the output delay assertions for both setup and hold analysis are removed.
`pin_or_port_list`	Specifies a single or multiple pins or ports for which the output delay is reset. To specify multiple pins, enclose the list with curly braces (`{}`) and separate the pin information with white space.
`-rise`	Resets the output delay for the rising edge at the output port. If both `-rise` and `-fall` options are omitted, the output delay is reset on both the edges.

Related Information

`set_output_delay`

reset_path_exception

```
reset_path_exception
     [-type {false_path | cycle_addition | path_delay_constraint}]
     [{-from | -rise_from | -fall_from} from_list]
     [{-through | -rise_through | -fall_through} through_list]
     [{-to | -rise_to | -fall_to} to_list]
     [-early | -late]
     [-all]
```

Removes any previously set path exceptions (or path exception of the specified type) for the given paths. To remove all path exceptions, use the `-all` option with the `reset_path_exception` command.

Parameters

`-all`	Removes all path exceptions. This option cannot be used with the `-from`, `-through` and `-to` options.
`-early | -late`	Resets only the early (or late) paths. *Default*: Both `-early` and `-late` are reset.
`{-from | -rise_from | -fall_from}` *`from_list`*	Specifies pins at the start of the path exceptions that are to be reset.
`{-through | -rise_through | -fall_through}` *`through_list`*	Specifies the pins that the path to be reset goes through.
`{-to | -rise_to | -fall_to}` *`to_list`*	Specifies pins at the end of the path exceptions that are to be reset.
`-type {false_path | cycle_addition | path_delay_constraint}`	Removes only path exceptions of the specified type. *Default*: All path exception assertions are reset.

Examples

- In the example below the rising path from R1 to R2 has three different path exceptions asserted on it - they are listed in priority from lowest to highest. As shown in the reports below, the false path is the highest precedence and so no path is reported.

```
set_multicycle_path 2 -from R1 -rise_to R2
set_max_delay       3 -from R1 -rise_to R2
set_false_path        -from R1 -rise_to R2
report_timing -from R1 -rise_to R2
```

The following information is displayed:

```
No constrained timing paths with given description found.
Paths may be unconstrained (try '-unconstrained' option) or may not exist.


report_path_exceptions
```

The following report is displayed:

```
+---------------------------------+
|  From |   To   |      Late      |
|-------+--------+----------------|
| R1/CK | ^ R2/D | delay 3 ignored |
| R1/CK | ^ R2/D | add 1 ignored   |
| R1/CK | ^ R2/D | false           |
+---------------------------------+
```

- In this example, the false path exception is removed from the path. This causes the max path delay assertion to become active - as it is the next highest precedence.

```
reset_path_exception -rise_to R2/D -type false_path
report_path_exceptions
```

The following report is displayed:

```
+-------------------------------+
|  From |   To   |     Late     |
|-------+--------+--------------|
| R1/CK | ^ R2/D | delay 3      |
| R1/CK | ^ R2/D | add 1 ignored |
+-------------------------------+


report_timing -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Setup Check with Pin R2/CK
Endpoint:   R2/D (^) checked with  leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.000
- Setup                         0.077
+ Path Delay                    3.000 <---------
= Required Time                 2.923
- Arrival Time                  0.238
= Slack Time                    2.685
Clock Rise Edge                      0.000
= Beginpoint Arrival Time            0.000
+-------------------------------------------------------------------+
| Instance |     Arc     |  Cell   | Delay | Arrival | Required |
|          |             |         |       |  Time   |   Time   |
|----------+-------------+---------+-------+---------+----------|
|          | clk ^       |         |       |   0.000 |    2.685 |
| R1       | CK ^ -> Q ^ | DFFHQX1 | 0.170 |   0.170 |    2.855 |
```

```
| U2       | A ^ -> Y ^  | BUFX1   | 0.068 |   0.238 |    2.923 |
| R2       | D ^         | DFFHQX1 | 0.000 |   0.238 |    2.923 |
+-------------------------------------------------------------------+
```

- In this example, the `set_max_delay` exception is removed, leaving only the `set_multicycle_path` exception active.

```
reset_path_exception -rise_to R2/D -type path_delay_constraint
report_path_exceptions
```

The following report is displayed:

```
+------------------------+
|  From |   To   |  Late |
|-------+--------+-------|
| R1/CK | ^ R2/D | add 1 |
+------------------------+


report_timing -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Setup Check with Pin R2/CK
Endpoint:   R2/D (^) checked with  leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.000
- Setup                         0.077
+ Phase Shift                  10.000
+ Cycle Adjustment             10.000  <----- Multi-Cycle timing
= Required Time                19.923
- Arrival Time                  0.238
= Slack Time                   19.685
Clock Rise Edge                      0.000
= Beginpoint Arrival Time            0.000
+-------------------------------------------------------------------+
| Instance |     Arc     |  Cell   | Delay | Arrival | Required |
|          |             |         |       |  Time   |   Time   |
|----------+-------------+---------+-------+---------+----------|
|          | clk ^       |         |       |   0.000 |   19.685 |
| R1       | CK ^ -> Q ^ | DFFHQX1 | 0.170 |   0.170 |   19.855 |
| U2       | A ^ -> Y ^  | BUFX1   | 0.068 |   0.238 |   19.923 |
| R2       | D ^         | DFFHQX1 | 0.000 |   0.238 |   19.923 |
+-------------------------------------------------------------------+
```

- In this example, the multi-cycle path assertion is removed, and the analysis reverts back to single-cycle timing

```
reset_path_exception -rise_to R2/D -type cycle_addition
report_path_exceptions
```

None reported

```
report_timing -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Setup Check with Pin R2/CK
Endpoint:   R2/D (^) checked with  leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.000
```

```
- Setup                         0.077
+ Phase Shift                  10.000 <--- Single-cycle timing
= Required Time                 9.923
- Arrival Time                  0.238
= Slack Time                    9.685
Clock Rise Edge                      0.000
= Beginpoint Arrival Time            0.000
+-------------------------------------------------------------------+
| Instance |     Arc     |  Cell   | Delay | Arrival | Required |
|          |             |         |       |  Time   |   Time   |
|----------+-------------+---------+-------+---------+----------|
|          | clk ^       |         |       |   0.000 |    9.685 |
| R1       | CK ^ -> Q ^ | DFFHQX1 | 0.170 |   0.170 |    9.855 |
| U2       | A ^ -> Y ^  | BUFX1   | 0.068 |   0.238 |    9.923 |
| R2       | D ^         | DFFHQX1 | 0.000 |   0.238 |    9.923 |
+-------------------------------------------------------------------+
```

- The example below shows how the `-early` and `-late` options can be used to control the effect of issuing `reset_path_exception`. The example will issue multi-cycle path constraints against both setup and hold modes, and then reset only the setup side.

```
set_multicycle_path 3 -setup -rise_from [get_clocks PH1] -rise_through U2/A
set_multicycle_path 2 -hold  -rise_from [get_clocks PH1] -rise_through U2/A


report_timing -late -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Setup Check with Pin R2/CK
Endpoint:   R2/D (^) checked with  leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.000
- Setup                         0.077
+ Phase Shift                  10.000
+ Cycle Adjustment             20.000  <-- 3 cycles
= Required Time                29.923
- Arrival Time                  0.238
= Slack Time                   29.685
Clock Rise Edge                      0.000
= Beginpoint Arrival Time            0.000
+-------------------------------------------------------------------+
| Instance |     Arc     |  Cell   | Delay | Arrival | Required |
|          |             |         |       |  Time   |   Time   |
|----------+-------------+---------+-------+---------+----------|
|          | clk ^       |         |       |   0.000 |   29.685 |
| R1       | CK ^ -> Q ^ | DFFHQX1 | 0.170 |   0.170 |   29.855 |
| U2       | A ^ -> Y ^  | BUFX1   | 0.068 |   0.238 |   29.923 |
| R2       | D ^         | DFFHQX1 | 0.000 |   0.238 |   29.923 |
+-------------------------------------------------------------------+



report_timing -early -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Hold Check with Pin R2/CK
Endpoint:   R2/D (^) checked with  leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by  leading edge of 'PH1'
```

```
Other End Arrival Time          0.000
+ Hold                         -0.038
+ Phase Shift                   0.000
- Cycle Adjustment              0.000 <- Adjusted 2 cycles
= Required Time                -0.038
Arrival Time                  0.238
Slack Time                    0.276
Clock Rise Edge                      0.000
= Beginpoint Arrival Time            0.000
+-------------------------------------------------------------------+
| Instance |     Arc     |  Cell   | Delay | Arrival | Required |
|          |             |         |       |  Time   |   Time   |
|----------+-------------+---------+-------+---------+----------|
|          | clk ^       |         |       |   0.000 |   -0.276 |
| R1       | CK ^ -> Q ^ | DFFHQX1 | 0.170 |   0.170 |   -0.106 |
| U2       | A ^ -> Y ^  | BUFX1   | 0.068 |   0.238 |   -0.038 |
| R2       | D ^         | DFFHQX1 | 0.000 |   0.238 |   -0.038 |
+-------------------------------------------------------------------+
```

```
reset_path_exception -late -rise_from [get_clocks PH1] -rise_through U2/A
report_timing -late -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Setup Check with Pin R2/CK
Endpoint:   R2/D (^) checked with  leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.000
- Setup                         0.077
+ Phase Shift                  10.000 <-- Reset to single-cycle timing
= Required Time                 9.923
- Arrival Time                  0.238
= Slack Time                    9.685
Clock Rise Edge                      0.000
= Beginpoint Arrival Time            0.000
+-------------------------------------------------------------------+
| Instance |     Arc     |  Cell   | Delay | Arrival | Required |
|          |             |         |       |  Time   |   Time   |
|----------+-------------+---------+-------+---------+----------|
|          | clk ^       |         |       |   0.000 |    9.685 |
| R1       | CK ^ -> Q ^ | DFFHQX1 | 0.170 |   0.170 |    9.855 |
| U2       | A ^ -> Y ^  | BUFX1   | 0.068 |   0.238 |    9.923 |
| R2       | D ^         | DFFHQX1 | 0.000 |   0.238 |    9.923 |
+-------------------------------------------------------------------+
```

```
report_timing -early -from R1 -rise_to R2/D
```

The following report is displayed:

```
Path 1: MET Hold Check with Pin R2/CK
Endpoint:   R2/D (^) checked with  leading edge of 'PH1'
Beginpoint: R1/Q (^) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.000
+ Hold                         -0.038
+ Phase Shift                   0.000
- Cycle Adjustment             20.000 <-- Still 2 cycle adjustment
= Required Time               -20.038      (from new setup edges)
Arrival Time                   0.238
Slack Time                    20.276
```

```
Clock Rise Edge                         0.000
= Beginpoint Arrival Time               0.000
+-------------------------------------------------------------------+
| Instance |     Arc     |  Cell   | Delay | Arrival | Required |
|          |             |         |       |  Time   |   Time   |
|----------+-------------+---------+-------+---------+----------|
|          | clk ^       |         |       |   0.000 |  -20.276 |
| R1       | CK ^ -> Q ^ | DFFHQX1 | 0.170 |   0.170 |  -20.106 |
| U2       | A ^ -> Y ^  | BUFX1   | 0.068 |   0.238 |  -20.038 |
| R2       | D ^         | DFFHQX1 | 0.000 |   0.238 |  -20.038 |
+-------------------------------------------------------------------+
```

Related Information

set_false_path

reset_path_group

```
reset_path_group
     [-name group_name]
     [-all]
```

Removes the specified path groups or completely removes all the groups.

Parameters

`-all`	Removes all path groups.
`-name group_name`	Removes the named group completely.

Example

- The following command removes the group named GRPA:

```
reset_path_group -name GRPA
```

Related Information

- `group_path`

reset_propagated_clock

```
reset_propagated_clock
     pin_or_clock_list
```

Removes the propagated clock assertion for the specified clock waveforms or pins. The ideal or propagated status of clock endpoints in the fanout of *pin_or_clock_list* is determined by the remaining clock latency propagated clock assertions in the clock network.

Parameters

pin_or_clock_list	Specifies clocks or pins for which you want to reset the clock propagation mode to `ideal` mode. The *pin_or_clock_list* argument can be a collection, however no hierarchical pins are allowed in the list.

Example

- The following command sets the `CLK1` clock waveform to `propagated` mode:

  ```
  set_propagated_clock CLK1
  ```

- The following command resets the `CLK1` clock waveform back to `ideal` mode:

  ```
  reset_propagated_clock CLK1
  ```

- The following commands set the `PH1` clocks to propagated mode and then resets them:

  ```
  set_propagated_clock [get_clocks PH1]
  report_clocks
  ```

  # The following report is displayed:

  ```
  +-------------------------------------------------------------------+
  |                        Clock Descriptions                         |
  |-------------------------------------------------------------------|
  |       |        |         |       |       |       Attributes      |
  |-------+--------+---------+-------+-------+-----------------------|
  | Clock | Source | Period  |  Lead | Trail | Generated | Propagated |
  |  Name |        |         |       |       |           |            |
  |-------+--------+---------+-------+-------+-----------+------------|
  |  PH1  |  clk1  | 10.000 | 0.000 | 5.000 |     n     |     y      |
  |  PH2  |  clk2  | 10.000 | 0.000 | 5.000 |     n     |     n      |
  +-------------------------------------------------------------------+
  ```

  ```
  reset_propagated_clock [get_clocks PH1]
  report_clocks
  ```

  # The following report is displayed:

  ```
  +-------------------------------------------------------------------+
  |                        Clock Descriptions                         |
  ```

					Attributes	
Clock Name	Source	Period	Lead	Trail	Generated	Propagated
PH1	clk1	10.000	0.000	5.000	n	n
PH2	clk2	10.000	0.000	5.000	n	n

Related Information

get_propagated_clock

report_clocks

set_clock_latency

set_propagated_clock

set_annotated_check

```
set_annotated_check
     [-clock clock_check]
     [-cond sdf_condtion]
     check_value
     [-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low]
     [-rise]
     [-fall]
     [-min]
     [-max]
     [-from from_pins]
     [-to to_pins]
```

Annotates the setup, hold, recovery, or removal timing check value between two or more pins on a cell in the current design.

Parameters

check_value	Specifies the timing check value.
-clock *clock_check*	Specifies clock rising or falling. Default: Sets checks for both clock rise and fall. Use the *clock_check* argument to specify rise or fall.
-cond *sdf_condition*	Specifies the condition only if the library has a condition attached to the specified timing check arc. The *sdf_expression* must match the syntax of the condition specified in the library.
-fall	Specifies the data fall transition for the timing check. Default: Sets both -rise and -fall.
-from *from_pins*	Specifies start points of the timing arcs for which checks are annotated. Use the *from_pins* argument to specify a list of leaf cell clock pins.
-max	Specifies the maximum timing check for both data rise and data fall transitions. Use this option only if the design uses minimum and maximum (min_max) operating conditions.
-min	Specifies the minimum timing check for both data rise and data fall transitions. Use this option only if the design uses minimum and maximum (min-max) operating conditions.

`-rise`	Specifies the data rise transition for the timing check. Default: Sets both rise and fall.
`-setup | -hold | -recovery | -removal | -nochange_high | -nochange_low`	Specifies the timing check type. Make sure there is a corresponding timing arc between the *`from_pins`* and the *`to_pins`* arguments.
`-to` *`to_pins`*	Specifies endpoints of the timing arcs for which checks are annotated. Use the *`to_pins`* argument to specify a list of leaf cell clock pins.

Examples

- The following commands annotates a `nochange check` between data low and clock low. The nochange margin before the clock is `1.3` and the nochange margin after the clock is `2.4`:

```
set_annotated_check -nochange_low 1.3 -clock fall -fall -from {I1/CP} -to {I1/EN}
set_annotated_check -nochange_low 2.4 -clock fall -rise -from {I1/CP} -to {I1/EN}
```

- The following command annotates a `2.0` setup time between the `CP` clock pin and the `A` data pin of the `instancename` instance:

```
set_annotated_check -setup 2.0 from instancename/CP -to instancename/A
```

set_annotated_delay

```
set_annotated_delay
     [-net | -cell]
     [-rise | -fall]
     [-pvt {min |typ | max}]
     [-from from_pins]
     [-to to_pins]
     [-sdf_cond expression]
     [-delta_only]
     delay_value
```

Annotates delay to timing arcs. If you specify the `set_annotated_delay` command without any parameters, the delay value specified is assumed to include both the base delay and any delta (cross-talk) delay.

Parameters

delay_value	Specifies the delay value to be annotated to the selected timing arcs.
-delta_only	Populates the incremental delay slot with the delay value, so that you can differentiate between the base net delay and a delay offset that is due to crosstalk. The software reports the incremental (or delta) delay value separately in the timing report using `report_timing -format incr_delay`, and maintains the separate value for CPPR calculations. If the base delay was previously calculated or annotated from the SDF, the `-delta_only` parameter adds the specified delay to the existing base delay. **Note:** This parameter only can be used with the `-net` parameter; you cannot specify it with the `-cell` parameter.
-from *from_pins*	Specifies timing arcs to which the delay applies. The selected timing arcs are those that have their source pins specified using this option and the sink pins specified using the `-to` option. If this option is not specified, only timing arcs whose sink pins are specified using the `-to` option are selected.

`-net | -cell` Specifies whether the delay should be annotated to cell arcs or net arcs. The tool automatically detects if the arcs specified with the `-from` and `-to` options are cell arcs or net arcs. This option is ignored if it conflicts with the `-from` and `-to` options. Use this option if the pin specified by the `-from` option or the `-to` option is a bidirectional pin. See Figure 30-4.

`-pvt {min | typ | max}`

Specifies the PVT corner to which the delay applies.

Default: The delay applies to all three PVT corners.

`-rise | -fall` Specifies the transition to which the delay applies. For cell arc, this is the output transition.

Default: The delay applies to both transitions.

`-sdf_cond` *`expression`*

Specifies the sdf condition of the selected timing arcs.
Default: The delay applies to all the timing arcs selected using the `-from` and the `-to` options. The expression is ignored for net arcs.

Note: Using the `set_annotated_delay` command without `sdf_cond` option will overwrite all previously specified annotated delays with the `sdf_cond` option. So, annotate delays without the condition first before annotating delays with the condition.

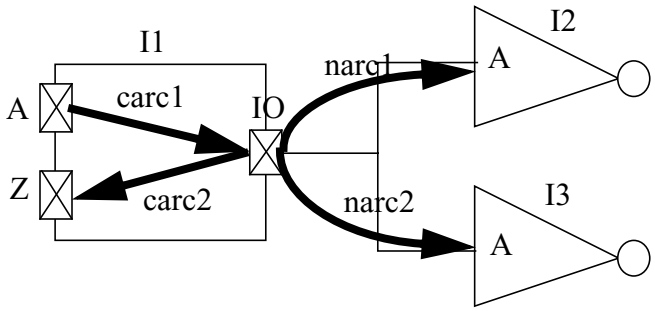
`-to` *`to_pins`* Specifies timing arcs to which the delay applies. The selected timing arcs are those that have their sink pins specified using this option and the source pins specified using the `-from` option. If this option is not specified, only timing arcs whose source pins are specified using the `-from` option are selected.

Examples

- The following command affects three timing arcs: the cell arc `carc2` and the net arcs `narc1` and `narc2` as shown in Figure 30-4:

  ```
  set_annotated_delay 3.0 -from {I1/IO}
  ```

Figure 30-4 Example Figure to Show How to Use the -cell and -net Options



- Use the -`cell` option to select the cell arc only:

  ```
  set_annotated_delay 3.0 -cell -from {I1/IO}
  ```

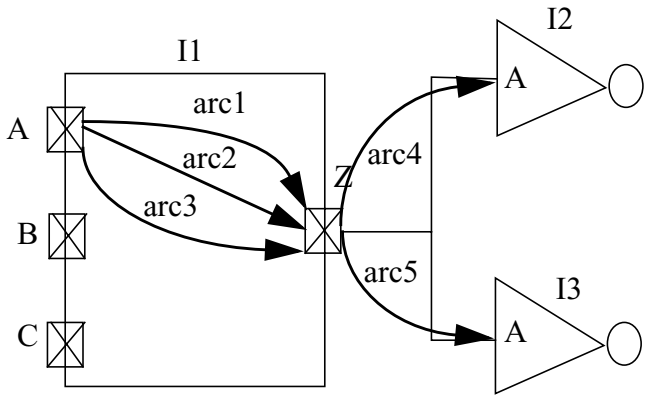
- Use the -`net` arc option to select net arcs only:

  ```
  set_annotated_delay 3.0 -net -from {I1/IO}
  ```

- This TLF description is used for the following examples:

```
CELL (NAND3
. . . .
// arc 1
Path( A => Z 10 01 DELAY(ioDelayRiseModel0) SLEW(SlopeRiseModel0) )
Path( A => Z 01 10 DELAY(ioDelayFallModel0) SLEW(SlopeFallModel0) )
// arc2
Path( A => Z 10  01 COND(B & C) SDF_COND(B & C) DELAY(ioDelayRiseModel1)
SLEW(SlopeRiseModel1) )
Path( A => Z 01 10 COND(B & C) SDF_COND(B & C) DELAY(ioDelayFallModel1)
SLEW(SlopeFallModel1) )
arc3
Path( A => Z 10 01 COND(~B | ~C) SDF_COND(!B | !C) DELAY(ioDelayRiseModel2)
SLEW(SlopeRiseModel2) )
Path( A => Z 01 10 COND(~B | ~C) SDF_COND(!B | !C) DELAY(ioDelayFallModel2)
SLEW(SlopeFallModel2) )
. . .
}
```

Figure 30-5 Example Figure to Show How to Use the -from and -to Options



- The following command annotates a delay of `3.0` to arcs `arc1`, `arc2`, and `arc3` for the rise and fall transition at the pin I1/Z, for the three pvt corners `{min typ max}`:

```
set_annotated_delay  3.0 -from A -to Z
```

- The following command annotates a delay of 2.5 to net arc `arc4` for both the rising and the falling transition and for the three pvt corners `{min typ max}`:

```
set_annotated_delay 2.5 -from I1/Z -to I2/A
```

- The following command annotates a delay of 1.5 to `arc2` for the rising transition and the min pvt corner:

```
set_annotated_delay 1.5 -from I1/A -to I1/Z -rise -pvt min -sdf_cond “B & C”
```

- The following commands annotate a delay of 3.0 to `arc1` and `arc2`, and a delay of 2.4 to the `arc3`:

```
set_annotated_delay 3.0 -from I1/A -to I1/Z
set_annotated_delay 2.4 -from I1/A -to I1/Z -sdf_cond {!B | !C}
```

- The following commands first annotate a delay of 3.0 to `arc1`, `arc2` and `arc3`, then overwrite the value set by the first `set_annotated_delay` command:

```
set_annotated_delay 2.4 -from I1/A -to I1/Z -sdf_cond {!B | !C}
set_annotated_delay 3.0 -from I1/A -to I1/Z
```

Related Commands

`set_annotated_check`

set_annotated_transition

```
set_annotated_transition
     [-rise | -fall]
     [-min | -max]
     slew_value
     pin_list
```

Sets the transition time to be annotated on specified ports or pins in the current design. The transition time that you specify using this command overrides the transition time calculated internally within the software. You can use this command to set the transition time of internal cell instance pins.

Parameters

`-min | -max`	Specifies whether the transition value is set for minimum or maximum transition time. *Default:* Transition value is set for both transition times.
`pin_list`	Specifies a list of pins or ports to be annotated with the transition time that you specify using the *`slew_value`* parameter.
`-rise | -fall`	Specifies whether the transition value is set for rising or falling transition time. *Default*: Transition value is set for both transitions.
`slew_value`	Specifies the transition time to be annotated to the selected pins or ports.

Examples

- The following command annotates a transition time value of 0.6 on port A:

  ```
  set_annotated_transition 0.6 [get_ports A]
  ```

- The following command annotates a transition time value of 0.7 on pin I2/A:

  ```
  set_annotated_transition 0.7 [get_pins I2/A]
  ```

- The following command annotates a transition value of 0.7 on pin I2/A for rising transition time:

  ```
  set_annotated_transition -rise 0.7 [get_pins I2/A]
  ```

- The following command annotates a transition value of 0.7 on pin I2/A for rising transition time and minimum pvt corner:

```
set_annotated_transition -rise -min 0.7 [get_pins I2/A]
```

Related Commands

set_input_transition

set_case_analysis

```
set_case_analysis
     {0 | 1 | zero | one | rising | falling | rise | fall}
     list_of_ports_or_pins
```

Sets constant or transitional values to a list of pins or ports for use by the timing engine. The specified constants or transitional values are valid only during timing analysis and do not alter the netlist. If there is a conflict between the specified `set_case_analysis` value and the logical value from the netlist, the value from the `set_case_analysis` is used in timing analysis.

Parameters

`0 | 1 | zero | one` Specifies a logic value of `0` or `1`, which is propagated through combinational logic to disable or enable parts of logic.

`rising | falling | rise | fall`

Specifies a valid transition value to be considered by the timing engine. For example, if a rising value is specified at a certain pin or port, only the rising signal transition from the pin or port is considered for timing analysis and the opposite transition type (falling) is ignored.

`list_of_ports_or_pins`

Specifies a list of ports or pins on which to apply the constant or transitional value. Specify multiple pins by enclosing them with curly braces `{}`and separating the pin names with white space. The *`list_of_ports_or_pins`* argument can be a collection.

Examples

- The following command sets the value of pin `i4/A` to `1` for use by the timing engine:

  ```
  set_case_analysis 1 i4/A
  ```

- The following example shows that the pins `P1/P2/B` and `P1/P3/CI` are considered only for a rising transition. The falling transition on these pins is ignored:

  ```
  set_case_analysis rising {P1/P2/B P1/P3/CI}
  ```

Related Information

check_timing -type constant_collision

report_case_analysis

report_timing

set_disable_timing

set_false_path

set_clock_gating_check

```
set_clock_gating_check
     [-high | low]
     [-rise | -fall]
     [-setup setup_value]
     [-hold hold_value]
     object_list
```

Specifies or overrides the default setup and hold values for clock gating checks.

Parameters

`-high | -low`	Specifies that the non-controlling value of the clock is high or low. Timing analysis performs the check on the non-controlling value of the clock. By default, the software determines the non-controlling value of the clock using information from the gate's logic. For example, for OR gates the non-controlling value is low. For complex gates, such as XOR and MUX, the clock is never controlling and it does not perform checks unless you specify either the `-high` option or the `-low` option. The specified value takes precedence over the default values. This option can only be used with pins or instances. **Note**: The clock pin value is controlling if it causes the gate output to be a constant (`0` or `1`).
`-hold` *`hold_value`*	Specifies the hold value. The default is `0.0`.
`object_list`	Specifies a list of clocks, instances, or pins. If a clock is specified, the check applies to all the clock gating gates driven by this clock. If an instance is specified, the check applies to all input pins of the instance. If the *`object_list`* argument is not specified, the check applies to all the clock gating checks in the design.
`-rise | -fall`	Applies the specified setup (or hold) value to the rising (or falling) signals on the specified pins. If the clock is specified or the *`object_list`* variable is not specified, the value applies to rising or falling, setup or hold. If neither option is specified, the default is both rising and falling.
`-setup` *`setup_value`*	Specifies the setup value. The default is `0.0`.

Note: The clock gating constraint is not propagated along the clock tree.

The setup check is performed with respect to the edge of the clock signal that changes the state of the clock pin from controlling to non-controlling. For example, for AND and NAND gates, the setup check is performed with respect to the rising edge of the clock input. For OR and NOR gates, the setup check is performed with respect to falling edge of the clock input.

The hold check is performed with respect to the edge of the clock signal that changes the state of the clock pin from non-controlling to controlling. For example, for AND and NAND gates, the hold check is performed with respect to the falling edge of the clock input. For OR and NOR gates, the hold check is performed with respect to rising edge of the clock input.

If the gate logic in unknown, setup and hold checks are performed with respect to both the rising edge and the falling edge of the clock signal and the timing analyzer uses the worst case values.

Priority Rules for Determining Setup or Hold Values

Timing analysis uses the following priority rules to determine the setup (or hold) value to use for a particular clock gating check:

1. Use the clock gating check assertion on the data pin if it exists
2. Otherwise, use the clock gating check assertion on the clock pin if it exists
3. Otherwise, use the clock gating check assertion on the clock waveform for the clock pin if it exists
4. Otherwise, use the global clock gating check assertion if it exists
5. If no clock gating check assertion is present, use the default setup (or hold) value of `0.0`.

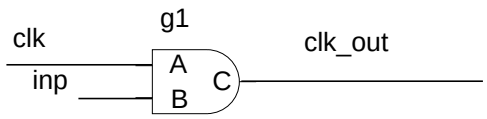
These rules are demonstrated in Example 30-2.

Note: This command has no effect on a clock gating check defined as a non-sequential setup or hold check in the library and reported as `ClockGatingSetup` or `ClockGatingHold`.

Examples

- Example 30-2 shows the priority rules for multiple clock gating check constraints for the gated clock shown in Figure 30-6.

Figure 30-6 Gated Clock



Example 30-2 Tcl Script for set_clock_gating_check

```
# constraints:
# clock
create_clock -name CLK1 -period 12 -waveform {0 6} clk
...
# clock-gating timing check assertions
# on ideal clock waveform
set_clock_gating_check -setup 0.44 [get_clocks {CLK1}]
# on clock pin
set_clock_gating_check -hold 0.11 [get_pins {g1/A}]
#on data pin
set_clock_gating_check -setup 0.66 -rise [get_pins {g1/B}]
set_clock_gating_check -hold 0.22 -fall [get_pins {g1/B}]
```

Given the previous constraints, the following timing margins are used for slack calculation:

- `SETUP g1/B rise: 0.66` — from assertion on data pin `g1/B` (rule 1)
- `SETUP g1/B fall: 0.44` — from assertion on ideal clock `CLK1` (rule 3)
- `HOLD g1/B rise: 0.11` — from assertion on clock pin `g1/A` (rule 2)
- `HOLD g1/B fall: 0.22` — from assertion on data pin `g1/B` (rule 1)

set_clock_groups

```
set_clock_groups
     [-name name]
     [ [-logically_exclusive] | [-physically_exclusive]
          | [[-asynchronous][-allow_paths]]
     [-group clock_list]
```

Defines clock groups with specified clock definitions. The software assigns all clocks that are defined after you use the set_clock_groups command to a default clock group. The software does not apply clock groupings on a master clock to the generated child clocks

If there is an overlap in the clock group specification, the software uses the following precedence to resolve the behavior:

1. Physically exclusive
2. Asynchronous
3. Logically exclusive

The software drops any explicit false path assertions between clocks that you have set as either physically or logically exclusive or asynchronous using the set_clock_groups command.

An alternative to setting the clocks as physically or logically exclusive using the set_clock_groups command is to use the multi-mode mechanism. Generally in an implementation flow, sequential setting of active clocks cannot be used. For concurrent analysis of different modes, you can use an multi-mode based approach to drive IPO rather than using the set_clock_groups command. For more information on multi-mode multi-corner analysis, see *Performing Multi-Mode Multi-Corner Timing Analysis and Optimization* chapter in the *Encounter User Guide*.

Parameters

-allow_paths	Allows paths between asynchronous clock groups to be traced in timing analysis. These paths are blocked by default. When you use this option, the software still considers the paths to be asynchronous for signal integrity considerations. Signal integrity does not support clock groups. Use the `-allow_paths` parameter with the `-asynchronous` parameter.

`-asynchronous` Defines clock group with asynchronous clocks. Asynchronous clocks have no specified phase relationship. The software considers asynchronous nets to have infinite timing window overlap for signal integrity considerations and considers these as false paths for timing analysis. Signal integrity does not support clock groups. Use the `-allow_paths` parameter to trace these paths in timing analysis.

`-group` *`clock_list`* Defines a set of clocks that are exclusive or asynchronous to the clocks defined in all other groups. If you define only one group, then all clocks not in the group implicitly belong to a default clock group. The software automatically adds any new clocks that you define after the `set_clock_groups` command to this default clock group.

`-logically_exclusive`

Defines clock group with clocks that do not have any functional paths between them, but coexist in the design at the same time. The software considers aggressor interactions between these clocks for signal integrity considerations, but considers these as false paths for timing analysis. Signal integrity does not support clock groups.

Example: clocks selected via a clock mux select

`-physically_exclusive`

Defines clock group with clocks that cannot exist in the design at the same time. The software does not consider aggressor interactions between clock groups that are physically exclusive for signal integrity considerations. Signal integrity does not support clock groups. The software considers these as false paths for timing analysis.

Example: multiple clocks defined on the same source pin

Examples

- The following commands assert two different clocks on the same clock root port, and uses set_clock_group to isolate them from each other:

```
create_clock -name SYSCLK -period 10 [get_ports sclk]
create_clock -name SCANCLK -period 100 -add [get_ports sclk]
set_clock_groups -name PHY_EX_SCLK -physically_exclusive
-group {SYSCLK} -group {SCANCLK}
```

- The following command declares a single bus clock to be asynchronous to all of the other on board clocks in the design (which remain in the default clock group):

```
set_clock_groups -name ASYNCH_BUS -asynchronous -group {BUSCLK}
```

set_clock_latency

```
set_clock_latency
     [-source [-early | -late]]
     [-rise | -fall]
     [-min | -max]
     latency
     pin_or_clock_list
```

Specifies ideal internal clock latency and external clock arrival delay.

Clock latency has two components:

- Source latency:
  - Specified using the `-source` parameter and is the delay from the external clock generator to the clock port of the design.
  - Applied when the clock is either ideal or propagated.
  - `set_clock_latency` set on a clock port has higher precedence than `set_clock_latency` set on a clock waveform.
  - Specified on clock waveforms with both the `set_input_delay` command and the `set_output_delay` command.
- Network latency (clock tree delay):
  - Default latency type and is the delay from the clock port to the register.
  - Applied only when the clock is ideal. When the clock is propagated, this delay is replaced by the actual clock tree delays.
  - Specified on a clock waveform or a clock port is ignored if the clock is propagated.
  - `set_clock_latency` set on a timing pin has higher precedence than `set_clock_latency` set on a waveform.
  - If there are multiple `set_clock_latency` and `set_propagated_clock` assertions between the clock waveform, clock root and clock endpoint, the closest assertion to the clock endpoint takes precedence.
  - Specified on internal circuit pins or hierarchical ports and used in the fanout cone of the network latency pin, when a clock is ideal.

Use the `set_clock_latency` command in two ways:

- To specify clock latency for a clock waveform:
  - Use the following syntax for source latency associated with a clock signal:

```
set_clock_latency [-source] [-early | -late] [-min | -max] latency
pin_or_clock_list
```

- Use the following syntax for network latency associated with a clock signal:

  ```
  set_clock_latency [-min | -max] latency pin_or_clock_list
  ```

  You cannot specify the `-late` and `-early` parameters for network latency.

Specifying latency on a port or pin overrides any previous latency specified on the port pin or waveform. The *`pin_or_clock_list`* argument is the list of the clock waveform names as specified by the `create_clock` command.

- To specify clock latency on clock pins:
  - Use the following syntax for source delay on a clock port:

    ```
    set_clock_latency -source [-early | -late] [-rise | -fall]
    insertion_delay pin_or_clock_list
    ```

  - Use the following syntax for network delay on a clock pin:

    ```
    set_clock_latency [-min] [- max] [-rise | -fall]
    insertion_delay -pin pin_or_clock_list
    ```

When a clock is ideal, the last specification along a path is used. This is useful when you want to intentionally skew the clock tree and model the physical hierarchical clock tree.

Parameters

`-early | -late`	Specifies clock arrival time with respect to the early or the late time of the clock signal. In setup analysis, launch path is the late path and capture path is the early path. In hold analysis, launch path is the early path and the capture path is the late path. If neither parameter is specified, the default is both early and late. Use these parameters only with the `-source` parameter. You cannot use these parameters with network latency. Specify the clock latency at the given operating corner. **Note:** The `-min` and `-max` parameters refer to the operating corner and the `-early` and `-late` parameters refer to the clock arrival time at the source. Use the `-early` and `-late` parameters with the `-source` parameter to specify the four different clock latencies.
`latency`	Specifies the latency value, as a floating point.

pin_or_clock_list	Specifies a list of clock waveform names or a list of pins to associate with the clock latency. The pin can be a port or an instance pin. Setting source latency on a pin is ignored.
-min | -max	Specifies the clock latency for a particular operating corner. Apply the -min parameter to the minimum operating corner and apply the -max parameter to the maximum operating corner.
-rise | -fall	Specifies the -rise or -fall parameter to the waveform at register clock pins for ideal latency and to the waveform at the source for source latency. Ideal network latency does not change polarity when crossing negative unate arcs.
-source	Specifies a source latency. If you do not specify the -source parameter, network latency is applied.

Examples

- The following set of commands enable you to create a clock using the create_clock command and apply latency to the created clock using the set_clock_latency command:

```
create_clock CLK1 -period 10 [get_ports clkA] # Define the ideal clock CLK1
create_clock CLK2 -period 20 [get_ports clkA clkB clkD] # Define ideal clock CLK2
set_clock_latency -source 2  {clkB clkD} # Define port settings
set_clock_latency -rise 5 {clkB clkD}  # Define port settings
set_clock_latency -early -late -source CLK1  # Define waveform settings
```

Related Information

create_clock

set_clock_transition

set_clock_uncertainty

set_input_delay

set_output_delay

set_propagated_clock

reset_clock_latency

set_clock_sense

```
set_clock_sense
     [-positive | -negative | -stop_propagation]
     [-clocks clock_list]
     pin_list
```

Selects which phase of the clock to filter at the specified point.

Parameters

`-clocks clock_list`	Specifies the list of clocks passing through the specified point to which the constraint applies.
`-negative`	Specifies that positive phase filtered clocks pass through pins in the *pin_list*.
pin_list	Specifies the list or collection of cell output pins. The unateness specified by the set_clock_sense command is applied to clocks propagating through the specified point.
`-positive`	Specifies that negative phase filtered clocks pass through pins in the *pin_list*.
`-stop_propagation`	Stops the propagation of specified clocks passing through the specified pins.

set_clock_transition

```
set_clock_transition
     [-rise | -fall]
     [-min |-max]
     slew_time
     clock_list
```

Specifies the slew time of register clock pins. This is useful for pre-layout static timing analysis before clock tree synthesis (CTS) is performed since net delays can be inaccurate with large fanout nets.

Use this command only with ideal clocks. In propagated mode, the software calculates slew times on register clock pins.

Parameters

clock_list	Specifies a list of clocks only with ideal mode in the design, which affects the slew times on all register clock pins in the transitive fanout of the specified clocks. Specified `set_clock_transition` values on register clock pins in the fanout of a clock with propagated latency are ignored.
`-rise | -fall`	Specifies the rising or falling edge of the register clock pins on which the slew time is asserted.
`-min | -max`	Specifies the clock slew time for minimum or maximum process corner slew time.
slew_time	Specifies the slew time (transition time) of the clock pins.

Example

- The following commands specify a `2.0` rise slew time and a `1.0` fall slew time for register clock pins clocked by `CLKA`:

```
set_clock_transition -rise 2.0 [get_clocks CLKA]
set_clock_transition -fall 1.0 [get_clocks CLKA]
```

Related Information

`report_clocks`

`set_clock_latency`

set_clock_uncertainty

set_input_delay

set_propagated_clock

set_clock_uncertainty

```
set_clock_uncertainty
     float
     [-setup | -hold]
     [-rise | -fall]
     {  {-from | -rise_from | -fall_from} clksig_from_list
        {-to | -rise_to | -fall_to} clksig_to_list
      | pin_or_clock_list
     }
```

Specifies the clock uncertainty (skew) on the clock network.

If you set multiple constraints with the `set_clock_uncertainty` command, the constraints set with the `-from*` and `-to*` parameters take precedence over constraints set with the *`pin_or_clock_list`* parameter.

When setting multiple constraints with the same parameters, the last command specified is applied, and the previous value is overwritten.

Parameters

`float`
Specifies the uncertainty value.

`-from | -rise_from | -fall_from` *`clksig_from_list`*

Applies the uncertainty value to the specified list of launching clocks.

Note: You must specify one of these parameters with one of the `-to`, `-rise_to`, or `-fall_to` parameters. You cannot use these parameters when *`pin_or_clock_list`* is specified.

`-fall_from`
Applies the uncertainty value only to the falling edge of the launching clocks.

`-from`
Applies the uncertainty value to both the rising and falling edges of the launching clocks.

`-rise_from`
Applies the uncertainty value only to the rising edge of the launching clocks.

pin_or_clock_list Specifies a list of clocks or pins for target uncertainty. The uncertainty value is applied to the registers clocked by the target waveform, or register pins in the fanout of the specified pin.

You can specify a list of clock names, or clock root pin or port names relative to the current module. You also can specify a collection of clocks, or clock root pins or ports.

Specifying a register or latch clock pin, or an arbitrary pin in the clock tree for *pin_or_clock_list* currently is not supported.

Note: You cannot use this parameter when any of the `-from` and `-to` parameter pairs are specified.

`-rise | -fall` Specifies the rising or falling edge at the capture clock pin on which the uncertainty is applied.

Default: Applies uncertainty on both edges

Note: You must use these parameters with the `-to` and `-from` parameters.

`-setup | -hold` Specifies whether the clock uncertainty applies to setup or hold checks.

Default: Applies to both setup and hold check

`-to | -rise_to | -fall_to` *clksig_to_list*

Applies the uncertainty value to the specified list of capture clocks.

Note: You must specify one of these parameters with one of the `-from`, `-rise_from`, or `-fall_from` parameters. You cannot use these parameters if *pin_or_clock_list* is specified.

`-fall_to` Applies the uncertainty value only to the falling edge of the capture clocks.

`-rise_to` Applies the uncertainty value only to the rising edge of the capture clocks.

`-to` Applies the uncertainty value to both the rising and falling edges of the capture clocks.

Examples

- The following command gives all paths starting from registers clocked by `clk1` to registers clocked by the rising edge of `clk2` a hold uncertainty of `2.0`:

  ```
  set_clock_uncertainty 2.0 -from clk1 -to clk2 -rise -hold
  ```

  In case of multiple constraints, the actual uncertainty applied by the software depends on which constraint matches the specific path in question.

- The following command sets an uncertainty on the `clk_in` clock root pin that is applied to all the register pins in the fanout of `clk_in`:

  ```
  set_clock_uncertainty 1.2 clk_in
  ```

- The following command specifies an inter-clock uncertainty of `0.2` between clock waveforms `clk1` and `clk2`:

  ```
  set_clock_uncertainty 0.2 -from clk1 -to clk2
  ```

- The following command specifies a target uncertainty of `0.5` for all registers clocked by the clock waveform `ideal_clk1`:

  ```
  set_clock_uncertainty 0.5 ideal_clk1
  ```

Related Information

`report_clocks`

`set_clock_latency`

`set_clock_transition`

`reset_clock_uncertainty`

set_data_check

```
set_data_check
     [{-from | -rise_from | -fall_from} pin_or_port_list]
     [{-to | -rise_to | -fall_to} pin_or_port_list]
     [-setup | -hold]
     [-clock clock_object]
     value
```

Sets data-to-data checks using the specified values of setup and hold time. Data-to data checks are non-sequential checks that are designed to constrain a data signal with respect to another data signal.

In a normal sequential check, a data signal is constrained with respect to a clock signal. In these cases single-cycle timing is used - allowing one full clock cycle for the data transfer. In a non-sequential data-to-data check, there is no clock reference. Therefore zero-cycle timing rules are used. You can use the set_multicycle_path constraint to adjust the cycle accounting rules.

Parameters

`-clock` *`clock_object`*

Specifies the name of the clock that triggers the data event for the related (`-from`) pin.

In the case where there are multiple data events triggered by different clock phases arriving at the `-from` pin, the `-clock` parameter allows you to specify which data event should be used for the “related pin” side of the timing check.

`-from |-rise_from | -fall_from` *`pin_or_port_list`*

Specifies the pins or ports that are set as related pins for data-to-data checks.

The *pin_or_port_list* argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified.

Use only one `-from`, `-rise_from`, or `-fall_from` parameter per command.

By default, the `-from` parameter checks both the rising and the falling edges.

Using `-rise_from` parameter checks at the rising edge of the signal on the from pins.

Using `-fall_from` parameter checks at the falling edge of the signal on the from pins.

`-setup | -hold`

Specifies whether the data check value is for setup or hold check. If you do not specify either `-setup` or `-hold` option, the value applies to both setup and hold.

`-to | -rise_to | -fall_to` *pin_or_port_list*

Specifies the pins or ports that are set as constrained pins for data-to-data checks.

The *pin_or_port_list* argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified.

Use only one `-to`, `-rise_to`, or `-fall_to` parameter per command.

By default, the `-to` parameter constrains both the rising and the falling edges.

Using `-rise_to` parameter constrains the rising edge of the signal on the to pins.

Using `-fall_to` parameter constrains the falling edge of the signal on the to pins.

value

Specifies the value of the setup or hold time for the check.

set_disable_clock_gating_check

```
set_disable_clock_gating_check
     object_list
```

Disables clock gating checks for the specified instances or pins.

Parameters

object_list	Specifies the list of pins or instances that are to be disabled.

Examples

- The following command disables clock gating checks for the specified instance:

  ```
  set_disable_clock_gating_check [get_cells clockgate1]
  ```

set_disable_timing

```
set_disable_timing
     [-from pin_name -to pin_name]
     object_list
```

Disables timing propagation through the specified arcs.

Parameters

`-from` *`pin_name`* `-to` *`pin_name`*

Specifies the arc that is to be disabled. Specify the `-from` and `-to` options together.

You can specify the `-from` and `-to` parameters when the *`object_list`* contains instance or library cell objects. Do not specify the `-from` and `-to` parameters when the *`object_list`* includes instance pin or library cell pin objects.

The `-from` and `-to` options are optional. If not specified, all cell arcs found in the instance or library cell objects are disabled.

`object_list`

Specifies the instance, instance pin, library cell, or library cell pin objects.

If the *`object_list`* includes instance pin or library cell pin objects, all cell arcs attached to the specified pin objects are disabled.

If the *`object_list`* contains instance or library cell objects, and the `-from` and `-to` parameters are not specified, all arcs found in the specified instance or library cell objects are disabled.

Examples

- The following command disables the arcs that lead to the output pins of the `i1`, `i2`, and `i3` instances. Note that timing checks at the inputs of `i1`, `i2`, and `i3` are still active.

  ```
  set_disable_timing [get_cells {i1 i2 i3}]
  ```

- The following command disables the arc from pin `B` to pin `Z` in the `i1` and `i2` instances, and the arc from pin `A` to pin `Z` in the `i3` instance:

  ```
  set_disable_timing -from B -to Z [get_cells {i1 i2}]
  set_disable_timing -from A -to Z [get_cells {i3}]
  ```

- The following example disables timing check arcs from pin `CLK` to pin `D` of the `ff1` instance:

  ```
  set_disable_timing -from CLK -to D [get_cells ff1]
  ```

- The following example disables the outgoing cell arcs from instance input pin `i1/B`:

  ```
  set_disable_timing [get_pins i1/B]
  ```

- The following example disables all arcs found in any instances whose reference cell is `AN2`:

  ```
  set_disable_timing [get_lib_cells AN2]
  ```

Related commands

`report_timing`

set_dont_touch

```
set_dont_touch
     object_list
     [true | false]
```

Prevents the specified object from being modified during optimization.

When you specify a net, all the connecting instances are affected by the `set_dont_touch` command.

Parameters

`true | false`	Sets or removes the `set_dont_touch` attribute for the specified *`object_list`*. *Default:* `true`
`object_list`	Specifies the list of instances, nets, library cells, or submodules.

Examples

- The following command prevents the specified instance from being modified:

  ```
  set_dont_touch [get_cells {TWA/FF1}]
  ```

set_dont_touch_network

```
set_dont_touch_network
     object_list
```

Prevents the combinational path connected to the given clocks, pins, or ports from being modified.

Parameters

object_list	Specifies the list of clocks, pins, or ports.

Examples

- The following command prevents the combinational path connected to the specified pin from being modified:

  ```
  set_dont_touch_network [get_pins and1/z]
  ```

set_dont_use

```
set_dont_use
     [true | false]
     object_list
```

Prevents the specified library cells from being used during optimization.

Parameters

`true | false`	Sets or removes the set_dont_use attribute for the specified object list. Default: true
`object_list`	Specifies the list of library cells.

Examples

- The following command prevents the specified library cell from being used during optimization:

  ```
  set_dont_use [get_lib_cells {and2}]
  ```

set_drive

```
set_drive
     [-rise] [-fall]
     [-max] [-min]
     resistance_value
     port_list
```

Sets the drive resistance on the specified input and/or bidirectional ports of the top cell.

The drive resistance models the cell driving the port. This resistance is used to compute the load dependent delay and the slew at the specified ports. The load-dependent delay is added to the arrival times.

The load-dependent delay and the slew are computed by multiplying the drive resistance by the wire capacitance of the net connected to the port.

Use the `report_ports` command to view drive resistances asserted on ports.

Parameters

`-fall`	Specifies that the drive resistance is to be used for the falling transition at the specified ports. By default, if neither the `-rise` nor the `-fall` option are specified, the drive resistance applies to both the rising transition and the falling transition.
`-min`	Specifies that the drive resistance is to be used to compute the early path delay. By default, if neither the `-min` option nor the `-max` option is specified, the drive resistance applies to both the early path delay and the late path delay.
`-max`	Specifies that the drive resistance is to be used to compute the late path delay. By default, if neither the `-min` option nor the `-max` option is specified, the drive resistance applies to both the early path delay and the late path delay.
`port_list`	Specifies a list of input or bidirectional ports on which the drive resistance is set.
`resistance_value`	Specifies the value of the drive resistance. This value must be greater than or equal to zero. The drive resistance is in units found in the target technology library.

`-rise` Specifies that the drive resistance is to be used for the rising transitions at the specified ports. By default, if neither the `-rise` option nor the `-fall` option are specified, the drive resistance applies to both the rising transition and the falling transition.

Example

- The following command sets a drive resistance of 2.0 for the early path, the late path, the rising and the falling transitions at the `in2` and `in2` ports and shows how drive resistances are reported using the `report_ports` command as shown in Example 30-3:

```
set_drive 2.0 [get_ports {in1 in2}]
report_ports -type drive_resistance -pin [get_ports {in1 in2}]
```

Example 30-3 set_drive get_ports report_ports Command Report

```
+-------------------------------------------------------------------------+
|       |       |                  |         |     Early     |     Late      |
|-------+-------+------------------+---------+---------------+---------------|
|  Pin  |  Dir  |    Assertion     |  Clock  |  Rise |  Fall |  Rise |  Fall |
|  Name |       |                  |  Name   |       |       |       |       |
|-------+-------+------------------+---------+-------+-------+-------+-------|
|  in1  |  IN   | drive_resistance | *(D)(P) |  2.00 |  2.00 |  2.00 |  2.00 |
|  in2  |  IN   | drive_resistance | *(D)(P) |  2.00 |  2.00 |  2.00 |  2.00 |
+-------------------------------------------------------------------------+
```

Related Commands

`all_inputs`

`report_ports`

`set_driving_cell`

`set_input_transition`

`set_load`

set_driving_cell

```
set_driving_cell
     [-library library_name]
     [-lib_cell cell_name]
     [-pin pin_name]
     [-from_pin from_pin_name]
     [-rise | -fall]
     [-min | -max]
     [-input_transition_rise rise_slew_value]
     [-input_transition_fall fall_slew_value]
     port_list
```

Models the drive capability of an external driver connected to the input port. Use this command only for timing analysis. The command does not affect the electrical properties of the design, which means that the capacitance of the output pin of the driving cell does not add to the total capacitance of the input port.

To model the drive capability of the external driver, the timing analyzer computes an offset to the arrival time of an input, and changes the slew time used to compute the delay for the next cell. The timing analyzer computes the offset by subtracting the zero load delay from the total delay of the cell. The `set_driving_cell` command uses the offset value to identify a cell from the library to drive the specified input port(s).

This command overrides the `set_drive` command per port or per net, if it is the last command applied. Also, if the `set_driving_cell` command is set for a port, the `set_input_transition` command is ignored for that port. If you do not define drive using the `set_driving_cell` command, the software uses the default value defined in the configuration file. The default value is `0.1ps`.

Parameters

`-fall`
: Annotates only the falling drive assertion. The default is to annotate both rise and fall delays.

`-from_pin` *`from_pin_name`*
: Specifies the name of the input port of the driver cell that has an arc to the output port of the driver cell, which is connected to the input port of the design. This option is required only if there are multiple timing arcs in the driver cell from its input ports to its output ports, and the arcs from those pins have different drive characteristics.

  If you do not specify the `-from_pin` option, the default is to use the first timing arc found on the library cell.

`-input_transition_fall` *`fall_slew_value`*

Specifies the slew value for the falling signal at the input of the drive cell.

`-input_transition_rise` *`rise_slew_value`*

Specifies the slew value for the rising signal at the input of the drive cell.

`-lib_cell` *`cell_name`*

Specifies the name of the driver cell.

`-library` *`library_name`*

Specifies the name of the library that contains the driving cell.

`-max` Annotates only the late drive assertion. The default is to annotate both early and late.

`-min` Annotates only the early drive assertion. The default is to annotate both early and late.

`-pin` *`pin_name`* Specifies the name of the output port of the driver cell that drives the signal at the input port of the design. This option is required if there are multiple output ports on the driver. If there is only one output port on the driver cell, then using this option is not necessary.

`port_list` Specifies the list of ports driven by the driver cell.

`-rise` Specifies only the rising drive assertion is annotated.The default is to annotate both rise and fall delays.

Related Information

`all_inputs`

`report_ports`

`set_drive`

`set_input_transition`

set_false_path

```
set_false_path
     [-hold | -setup]
     [-rise]
     [-fall]
     [{-from | -rise_from | -fall_from} pin_list]
     [{-through | -rise_through | -fall_through} pin_list]
     [{-to | -rise_to | -fall_to} pin_list]
```

Identifies false paths in a design, and breaks or disables specific instance timing arcs in a design.

Important

To break combinational loops, use the `set_disable_timing` command instead of the `set_false_path` command.

The `set_false_path` command differs from the `set_disable_timing` command as follows:

- `set_false_path` command selectively disables the arrival time information depending on the precedence rules.
- `set_false_path` command does not affect constant values.
- `set_disable_timing` command physically snips a timing arc such that neither arrival times nor constant values propagate through it.
- `set_disable_timing` command is applied in all cases to valid timing arcs.

The `set_false_path -from fpin` command disables all timing paths whose source pin is an `fpin`. Similarly, the `set_false_path -to tpin` command disables all timing paths whose sink pin is a `tpin`. Using the `set_false_path -from fpin -to tpin` command disables all timing paths whose source is `fpin` and sink is `tpin`.

Specify a group of pins by enclosing the group with curly braces (`{}`) within one `set_false_path` command.

Tip

To set a false path for any path beginning at `p1` or `p2` and terminating at `t1` or `t2`, group the pins as follows:

```
set_false_path -from {p1 p2} -to {t1 t2}
```

Note: This method of pin grouping reduces the number of exceptions. Having large numbers of exceptions adversely impacts the run time of the timing analysis commands.

Parameters

`-fall`

Applies the assertion at the falling edge of the signal on the path endpoints.

`{-from | -rise_from | -fall_from}` *`pin_list`*

Specifies a list of timing paths start points. The *`pin_list`* argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. The *`pin_list`* argument can be a collection.

The valid start points are:

- Input port
- Bidirectional port
- Clock pin of sequential cell

  Clock pin of sequential cell is one of the following:

  - Structural clock pin that is part of latch or flip-flop.
  - Inferred clock pin in cells like macro cells. A pin is an inferred clock pin if it is associated with one of the following:
    - Reference end of setup check
    - Reference end of hold check
    - Reference end of removal check
    - Reference end of recovery check
    - Minimum period check
    - Minimum pulse width check
- Pin associated with `set_input_delay`

Specify one of the following:

`-from` Applies the false path constraint to all paths originating from the *`pin_list`* list. If a clock is specified as the `-from` object, all primary inputs and registers clocked by that clock are used as start points.

By default, the `-from` option applies the constraint to both the rising and the falling edges.

`-rise_from` Applies the false path constraint at the rising edge of the signal on the from pins.

`-fall_from` Applies the false path constraint at the falling edge of the signal on the from pins.

`-hold | -setup` Sets false paths for either setup or hold analysis.

Note: If you do not specify `-setup` or `-hold`, the software sets false paths for both setup and hold analysis.

`-rise` Applies the assertion at the rising edge of the signal on the path endpoints.

`{-through | -rise_through | -fall_through}` *`pin_list`*

Specifies a list of timing paths through points.

The *`pin_list`* argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. The *`pin_list`* argument can be a collection.

Specify one of the following:

`-through` When you use this parameter without using the `-from` or `-to` options, the false path constraint is set for all paths that go through the specified pins. When you use it with both the `-from` and `-to` options, any path starting at any pin on the `-from` list *and* going through any point on the `-through` pin list *and* going to any point on the `-to` list is affected by the constraint.

By default, the `-through` option applies the constraint to both the rising and the falling edges.

`-rise_through`

Applies the assertion at the rising edge of the signal on the through pins.

`-fall_through`

Applies the assertion at the falling edge of the signal on the through pins.

`{-to | -rise_to | -fall_to}` *`pin_list`*

Specifies a list of timing paths end points.

The *`pin_list`* argument can contain either object IDs or hierarchical names relative to the current module. The pins can be on intermediate hierarchical boundaries. The *`pin_list`* argument can be a collection.

The valid end points are:

- Output port
- Bidirectional port
- Data pin of sequential cell

  Data pin of sequential cell is one of the following:

  - Structural data pin that is part of latch or flip-flop.
  - Inferred data pin in cells like macro cells. A pin is an inferred data pin if it is associated with one of the following:
    - Signal end of setup check
    - Signal end of hold check
    - Signal end of non-sequential setup check
    - Signal end of non-sequential hold check
    - Signal end of removal check
    - Signal end of recovery check
- Pin associated with `set_external_delay`
- Data pin of clock gating cell

Select from one of the following:

`-to`	Applies the false path constraint to all paths ending in the `-to` pin list. If a clock is specified as the `-to` object, all primary outputs and registers clocked by that clock are used as endpoint. By default, the `-to` option applies the constraint to both the rising and the falling edges.
`-rise_to`	Applies the false path constraint at the rising edge of the signal on the to pins.
`-fall_to`	Applies the false path constraint at the falling edge of the signal on the to pins.

Examples

- The following command disables hold checking from `ff1` to `ff2`:

  ```
  set_false_path -hold -from ff1 -to ff2
  ```

- The following command sets all timing paths from `ff1/CK` to `D` input of `ff2` that pass through `u1/Y` and one or more of `u2/Y` and `u3/Y` as false paths:

  ```
  set_false_path -from ff1/CK -through {u1/Y} -through {u2/Y u3/Y} -to ff2/D
  ```

- The following command sets all timing paths from the rise of `ff1/CK` to `ff2/D` through one or more of `u1/Y` and `u4/Y` as false paths:

  ```
  set_false_path -rise_from ff1/CK -through {u1/Y u4/Y} -to ff2/D
  ```

Related Information

`set_disable_timing`

`set_max_delay`

`set_min_delay`

`set_multicycle_path`

set_fanout_load

```
set_fanout_load
     fanout_load
     port_list
```

Specifies the fanout load on a set of ports. Library instance pins may have fanout loads set in the technology library. Fanout does not have specific units so it is assumed that fanout as used by constraints is consistent with fanout in the technology libraries. If a pin or port has no fanout attribute, either from constraints or from the technology library, the fanout load is zero. The fanout load is used only for design rule checking; it does not affect timing.

Parameters

fanout_load	Specifies the fanout load value for the port or ports.
port_list	Specifies the list of ports for which the fanout load is applied.

Examples

- The following command sets the fanout load of port `out` to 2:
  ```
  set_fanout_load 2 out
  ```
- The following command sets the fanout load of ports `in1` and `in2` to 1:
  ```
  set_fanout_load 1 {in1 in2}
  ```
- The following command sets the fanout load of element 3 of port `out` to 12:
  ```
  set_fanout_load 12 { out[3] }
  ```
- The following command sets the fanout load of all the ports to 2:
  ```
  set_fanout_load 2 [get_ports *]
  ```

Related Information

`report_net`

`report_ports`

`set_max_fanout`

set_input_delay

```
set_input_delay
     [-clock clock_name]
     [-clock_fall]
     [-rise] [-fall]
     [-max] [-min]
     [-add_delay]
     [-network_latency_included] [-source_latency_included]
     [-reference_pin pin_name]
     [-level_sensitive]
     delay_value
     port_or_pin_list
```

Defines the arrival time relative to a clock edge on input ports or internal input pins. This input path delay models the delay from an external register to an input port of the module.

If the input delay is not specified on an input port, it is assumed to be zero.

Parameters

`-add_delay`	Adds delay information to the existing input delay. The additional delay information can be for multiple paths relative to different clocks or clock edges leading to an input port or internal input pins. If you do not use the `-add_delay` option, the software uses the last `set_input_delay` related to a pin when there are multiple `set_input_delay` statements for the same pin.
`-clock` *`clock_name`*	Specifies that the delay is relative to the rising edge of the given clock. If you do not specify the `-clock` parameter, the delay is relative to time zero. To specify a delay relative to the falling edge of the clock, use the `-clock_fall` option with the `-clock` option.
`-clock_fall`	Specifies that the delay is relative to the falling edge of the clock. You must specify the `-clock` parameter with this parameter. *Default:* Delay is relative to the rising edge.
`delay_value`	Specifies the value for the path delay.
`-fall`	Specifies that the input delay refers to the falling edge of the signal at specified ports or pins. If you do not specify the `-fall` option, the input delay applies to both the edges.

`-level_sensitive` Models input delay that is launched by a positive level sensitive latch.

Use the `-clock` parameter to model a positive level sensitive latch, or the `-clock_fall` parameter to model a negative level sensitive latch.

`-max` Specifies that the delay refers to the setup analysis. If you do not specify the `-max` parameter, the input delay applies to both setup and hold analysis.

`-min` Specifies that the delay refers to the hold analysis. If you do not specify the `-min` parameter, the input delay applies to both setup and hold analysis.

`-network_latency_included`

Specifies that the clock network latency should not be added to the input delay value. If you do not specify this option, the clock network latency of the related clock is added to the input delay value. This parameter has no effect if the specified clock is in propagated mode.

`port_or_pin_list` Specifies a list of input port or pin names. The *`port_or_pin_list`* argument can be a collection.

`-reference_pin` *`pin_name`*

Adds source and network latency of the specified reference pin to the input delay value in propagated mode. A reference pin is a leaf pin in the fanout cone of the clock source of the clock that you specify using the `-clock` parameter. You cannot use the `-reference_pin` parameter with the `-network_latency_included` or `-source_latency_included` parameters.

`-rise` Specifies that input delay refers to the rising edge of the signal at the specified ports or pins. If you do not specify the `-rise` option, the input delay applies to the both edges.

`-source_latency_included`

Specifies that the clock source latency should not be added to the input delay value.

Examples

- The following command sets an input delay of 1.0 for all inputs in the current design relative to the rising edge of the CLK1 clock:

```
set_input_delay -clock CLK1 1.0 [all_inputs]
```

- The following command sets an input delay of 2.0 for the input IN1 port relative to the falling edge of the CLK1 clock:

```
set_input_delay -clock CLK1 2.0 -clock_fall { IN1 }
```

- The following example shows an input IN1 port that is constrained by both the rising edge and the falling edge of the same CLK1 clock. The -add_delay option prevents the second set_input_delay statement from overwriting the first set_input_delay:

```
set_input_delay -clock CLK1 1.0 { IN1 }
set_input_delay -clock CLK1 -clock_fall -add_delay 2.0 { IN1 }
```

- When multiple -min and -max values are specified relative to the same clock using the -add_delay option at the same port or pin, the worst case value for each case is used. For the -min case the smallest value is retained, and for the -max case the largest value is retained. For example, the input delay value of 1.0 is stored for the shortest path and the value of 4.0 is stored for the longest path in the following example:

```
set_input_delay -min -clock CLK1 1.0 { IN1 }
set_input_delay -min -clock CLK1 2.0 -add_delay { IN1 }
set_input_delay -max -clock CLK1 3.0 -add_delay { IN1 }
set_input_delay -max -clock CLK1 4.0 -add_delay { IN1 }
```

Related Information

all_inputs

create_clock

report_ports

set_load

set_output_delay

set_input_transition

```
set_input_transition
     [-min | -max]
     [-rise |-fall ]
     transition_time
     port_list
```

Specifies the slew time for a port on the top level module. This command is useful only in specifying a hard ramp signal as if the port were driven by a very powerful driver. The command does not change the arrival time of an input, but changes the slew time used to compute the delay of the cell on the sink of a net. If you do not define transition using the `set_input_transition` command, the software uses the default input port transition defined in the configuration file. The default value is `0.1ps`. You can also use the `setInputTransitionDelay` command to override the default value.

Note: The `set_input_transition` command is ignored for a port if `set_driving_cell` or `set_drive` constraints have been set for that port.

Parameters

`-fall | -rise`	Specifies that the slew time should be applied to the rising edge or the falling edge of the signal (data or clock). If neither time is specified, the default is to use both.
`-min | -max`	Indicates whether the slew times are with respect to the early (hold) or the late (setup) time checks of the data signal. If neither option is specified, the default is to use both.
`port_list`	Specifies the list of ports for which the slew time is specified.
`transition_time`	Specifies the transition time.

Examples

- The following command specifies the slew time of `0.5` on port, `input3`.

  ```
  set_input_transition 0.5 input3
  ```

Related Commands

`report_ports`

`set_driving_cell`

set_drive

set_clock_transition

set_lib_pin

```
set_lib_pin
     collection_of_library_pins
     [-max_fanout value]
     [-max_transition value]
     [-max_capacitance value]
```

Overrides library pin design rule attributes that are specified in the Liberty timing library.

Note: Any library pin modifications made with the `set_lib_pin` command are valid only for the current Encounter session. You cannot save the modifications in the database, or restore them in another session.

Parameters

`collection_of_library_pins`

Overrides design rule attributes for the specified collection of library pins. Use the `get_lib_pins` command to generate the library pin collection.

`-max_capacitance` *`value`*

Specifies the new value for the maximum capacitance limit.

`-max_fanout` *`value`* Specifies the new value of the maximum fanout load limit.

`-max_transition` *`value`*

Specifies the new value of the maximum slew time limit (maximum transition).

Examples

- The following command changes the `max_fanout` constraint for all instances of the `BUFF1` cell from library `slow` to 10:

  ```
  set_lib_pin [get_lib_pins slow/BUFF1/Z] -max_fanout 10
  ```

- The following command uses a more extensive collection method to change the `max_capacitance` constraint value of all outputs of all flip-flop output pins to 0.050:

  ```
  set_lib_pin
      [filter_collection [get_lib_pins slow/FF*/*] “direction == out”]
      -max_capacitance 0.050
  ```

Related Commands

set_max_capacitance

set_max_fanout

set_max_transition

set_load

```
set_load
     [-max] [-min]
     [-pin_load] [-wire_load]
     capacitance_value
     port_net_list
```

Sets the specified capacitance on the specified ports or nets of the top cell. The capacitance that you specify using this command overrides the capacitance from wire load models and SPEF.

For ports, the asserted capacitance is either the external pin capacitance when the `-pin_load` option is specified, or the external wire capacitance when the `-wire_load` option is specified, or the total external capacitance when neither option is specified.

Use the `report_ports` command to view capacitance asserted ports. The pin load appears in the report as a `port_cap` in the assertion column and the wire load appears in the report as a `port_wire_cap` in the assertion column.

Parameters

capacitance_value	Specifies the value for the capacitance. The capacitance is in units from the target technology library.
-min	Specifies the minimum capacitance. The minimum capacitance is used to compute best-case delays in BcWc analysis mode and early path delays in OCV analysis mode.
-max	Specifies the maximum capacitance. The maximum capacitance is used to compute worst-case delays in BcWc analysis mode and late path delays in OCV mode. It is also used to check the maximum capacitance design rule.
port_net_list	Specifies a list of ports or nets in the top cell.
-pin_load	Specifies the external pin capacitance for a port. Use this option only for ports. This is the default option if neither the `-pin_load` option nor the `-wire_load` option is not specified.
-wire_load	Interprets the specified capacitance as the external wire capacitance of the port. Use this option only for ports. Use this option with the *port_net_list* argument if the list contains only ports.

Examples

- The following command sets an external minimum pin capacitance of 1.5 units on the `in1` `in2` ports:

  ```
  set_load 1.5 -min -pin_load [get_ports {in1 in2}]
  ```

- The following command sets an external minimum wire capacitance of 1.2 units on the `in1` `in2` ports:

  ```
  set_load 1.2 -min -wire_load [get_ports {in1 in2}]
  ```

- The following command sets an external maximum pin capacitance of 2.6 units on the `in1` `in2` ports:

  ```
  set_load 2.6 -max -pin_load [get_ports {in1 in2}]
  ```

- The following command sets an external maximum wire capacitance of 3.0 units on the `in1` `in2` ports:

  ```
  set_load 3.0 -max -wire_load [get_ports {in1 in2}]
  ```

- The following command reports the capacitances set on the ports, as shown in Example 30-4:

  ```
  report_ports -pin [get_ports {in1}]
  ```

Example 30-4 report_ports -pin [get_ports {in1}] Report

```
+-------------------------------------------------------------------------------+
|       |       |           |             |      Early      |      Late       |
|-------+-------+-----------+-------------+-----------------+-----------------|
|  Pin  |  Dir  | Assertion |  Clock Name |  Rise  |  Fall  |  Rise  |  Fall  |
|  Name |       |           |             |        |        |        |        |
|-------+-------+-----------+-------------+--------+--------+--------+--------|
|  in1  |  IN   |   input   | clock(D)(P) | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
+-------------------------------------------------------------------------------+
     +-----------------------------------------------------+
     |  Pin  |  Dir  |   Assertion   |         Value         |
     |  Name |       |               |                       |
     |-------+-------+---------------+-----------------------|
     |  in1  |  IN   |   port_cap    | ( 1.5000 :  : 2.6000 ) |
     |  in1  |  IN   | port_wire_cap | ( 1.2000 :  : 3.0000 ) |
     +-----------------------------------------------------+
```

Related commands

`all_outputs`

`report_ports`

`set_drive`

`set_resistance`

set_logic_one

```
set_logic_one
     ports_pins
```

Sets ports or pins to logic 1 in the design.

Parameters

ports_pins	Specifies the list of the ports or pins that are to be set to logic 1.

Examples

- The following command sets the specified port to logic 1:

  ```
  set_logic_one [get_ports {din[1]}]
  ```

set_logic_zero

```
set_logic_zero
     ports_pins
```

Sets ports or pins to logic 0 in the design.

Parameters

ports_pins	Specifies the list of the ports or pins that are to be set to logic 0.

Examples

- The following command sets the specified port to logic 0:

  ```
  set_logic_zero [get_ports {din[0]}]
  ```

set_max_capacitance

```
set_max_capacitance
     capacitance_limit
     port_or_module_list
```

Sets the maximum capacitance limit on the specified ports of the top cell.

When set on a port, the total capacitance of the net connected to that port must be less than or equal to the specified limit. Maximum capacitance limits also come from library objects, which is the default maximum capacitance limit of libraries and the maximum capacitance limit on library cell pins. Pins in the design inherit the limits from libraries.

Note: The limit set on the top cell applies to its ports. Thus, if maximum capacitance limits are set on the top cell and its ports, the most constraining limit will apply to the ports.

Parameters

capacitance_limit Specifies the value for the maximum capacitance limit. The limit must be greater than or equal to zero. The limit is in capacitance units from the current session.

port_or_module_list

Specifies the list of ports and modules on which the maximum capacitance limit is applied.

Examples

- The following example sets a maximum capacitance limit of `1.2` units on all the ports of the top cell and a maximum capacitance limit of `0.5` units on the port named `d_out`:

```
set_max_capacitance 1.2 [get_ports *]
set_max_capacitance 0.5 d_out
```

- The following example uses the `report ports` command to show the maximum capacitance limit set on ports, as shown in Example 30-5:

```
set_max_capacitance 0.9 [get_ports *]
report_ports -type port_cap_limit -pin [get_ports *
```

Example 30-5 Maximum Capacitance Limit Set on Ports Report

```
+-----------------------------------------------+
|  Pin  |  Dir  |   Assertion    |     Value     |
|  Name |       |                |               |
|-------+-------+----------------+---------------|
|  in1  |  IN   | port_cap_limit | (  : 0.9000 ) |
|  in2  |  IN   | port_cap_limit | (  : 0.9000 ) |
|  out1 |  OUT  | port_cap_limit | (  : 0.9000 ) |
|  out2 |  OUT  | port_cap_limit | (  : 0.9000 ) |
|  out3 |  OUT  | port_cap_limit | (  : 0.9000 ) |
|  out4 |  OUT  | port_cap_limit | (  : 0.9000 ) |
+-----------------------------------------------+
```

Related Commands

get_ports

report_ports

set_max_transition

set_max_fanout

set_max_delay

```
set_max_delay
     [{-from |-rise_from | -fall_from} pin_or_clock_list]
     [{-through | -rise_through | -fall_through} pin__or_clock_list]
     [{-to | -rise_to | -fall_to} pin_or_clock_list]
     [-rise]
     [-fall]
     value
```

Specifies a maximum delay for a timing path. The `set_max_delay` command takes precedence over other path exceptions specified for the same path, except for the `set_false_path` command.

Use either the `-from`, `-to`, or `-through` option with the `set_max_delay` command to identify the paths. Specifying both the `-from` and `-to` options identifies a path with specific starting and ending points.

When you specify a sequential cell name for starting point of the timing path, the valid path starts from the clk pin. This means the latency and delay from clk to output Q is also included in the delay. For path segmentation, you can specify the path to start from Q output of the register by adding /Q after the cell name. In this case, path from clk to Q is not considered in the calculation.

Use only one `-from`, `-rise_from`, or `-fall_from`, `-to`, `-rise_to`, or `-fall_to` parameter per command. You can use multiple `-through` parameters in the same command.

Parameters

`-fall` — Apply the assertion at the falling edge of the signal on the path endpoints.

`-from |-rise_from | -fall_from` *`pin_or_clock_list`*

Specifies a list of timing paths start points. The *pin_or_clock_list* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *pin_or_clock_list* argument can be a collection.

The valid start points are:

- Input port
- Bidirectional port
- Clock pin of sequential cell

  Clock pin of sequential cell is one of the following:

  - Structural clock pin that is part of latch or flip-flop.
  - Inferred clock pin in cells like macro cells. A pin is an inferred clock pin if it is associated with one of the following:
    - Reference end of setup check
    - Reference end of hold check
    - Reference end of removal check
    - Reference end of recovery check
    - Minimum period check
    - Minimum pulse width check
- Pin associated with `set_input_delay`

Specify one of the following:

`-from`
: Applies the maximum delay constraint to all paths originating from the *pin_or_clock_list* list. If a clock is specified as the `-from` object, all primary inputs and registers clocked by that clock are used as start points.

  By default, the `-from` option applies the constraint to both the rising and the falling edges.

`-rise_from` Applies the maximum delay constraint at the rising edge of the signal on the from pins.

`-fall_from` Applies the maximum delay constraint at the falling edge of the signal on the from pins.

`-rise` Applies the assertion at the rising edge of the signal on the path endpoints.

`-through | -rise_through | -fall_through` *`pin_or_clock_list`*

Specifies a list of timing paths through points.

The *`pin_or_clock_list`* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *`pin_or_clock_list`* argument can be a collection.

Specify one of the following:

`-through` When you use this parameter without using the `-from` or `-to` options, all paths that go through the specified pins have the specified delay. When you use it with both the `-from` and `-to` options, any path starting at any pin on the `-from` list *and* going through any point on the `-through` pin list *and* going to any point on the `-to` list is affected by the constraint.

By default, the `-through` option applies the constraint to both the rising and the falling edges.

`-rise_through` Applies the assertion at the rising edge of the signal on the through pins.

`-fall_through` Applies the assertion at the falling edge of the signal on the through pins.

`-to | -rise_to | -fall_to` *`pin_or_clock_list`*

Specifies a list of timing paths end points.

The *pin_or_clock_list* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *pin_or_clock_list* argument can be a collection.

The valid end points are:

- Output port
- Bidirectional port
- Data pin of sequential cell

  Data pin of sequential cell is one of the following:

  - Structural data pin that is part of latch or flip-flop.
  - Inferred data pin in cells like macro cells. A pin is an inferred data pin if it is associated with one of the following:
    - Signal end of setup check
    - Signal end of hold check
    - Signal end of non-sequential setup check
    - Signal end of non-sequential hold check
    - Signal end of removal check
    - Signal end of recovery check
- Pin associated with `set_external_delay`
- Data pin of clock gating cell

Select from one of the following:

`-to`
: Applies the maximum delay constraint to all paths ending in the `-to` pin list. If a clock is specified as the `-to` object, all primary outputs and registers clocked by that clock are used as endpoint.

  By default, the `-to` option applies the constraint to both the rising and the falling edges.

	`-rise_to`	Applies the maximum delay constraint at the rising edge of the signal on the to pins.
	`-fall_to`	Applies the maximum delay constraint at the falling edge of the signal on the to pins.
`value`	Specifies the maximum delay value.	

Example

- The following command specifies that all paths from `in1` to `out1` must have delays less than `10` units:

  ```
  set_max_delay 10 -from in1 -to out1
  ```

Related Commands

`create_clock`

`set_false_path`

`set_input_delay`

`set_min_delay`

`set_multicycle_path`

`set_output_delay`

set_max_fanout

```
set_max_fanout
     fanout_limit
     port_or_module_list
```

Sets the maximum fanout load limit constraint on the specified input ports of the top cell and the specified modules.

When set on a port, the fanout load of the net connected to the port must be less than or equal to the specified limit value to meet the constraint. The fanout load of a net is the sum of the fanout loads of the pins driven by the net. When set on a module, the limit applies to all the nets of that module. Libraries and library cell pins may also have the maximum fanout load limit. In this case the most constraining limit is applied to the nets.

Use the `report_ports` `-type fanout_load_limit` command to view the maximum fanout load limits set on ports of the top cell.

Parameters

`fanout_limit` Specifies the value of the maximum fanout load limit constraint. This is a float number and it must be greater than or equal to zero.

`port_or_module_list`

Specifies a list of input ports on the top cell or a list of modules.

Examples

- The following example sets a maximum fanout load limit of `5.0` units on all the ports of the top cell:

  ```
  set_max_fanout 5.0 [get_ports *]
  ```

- The following example shows a report of the maximum fanout load limit set on ports using the `report_ports` command, as shown in Example 30-6:

  ```
  set_max_fanout 5.0 [get_ports *]
  report_ports -type fanout_load_limit -pin [get_ports *]
  ```

Example 30-6 Maximum Fanout Load Limit Set On Ports Report

```
+-----------------------------------------------+
|  Pin  |  Dir  |   Assertion    |     Value     |
|  Name |       |                |               |
|-------+-------+----------------+---------------|
|  in1  |  IN   | port_cap_limit | (  : 0.9000 ) |
|  in2  |  IN   | port_cap_limit | (  : 0.9000 ) |
|  out1 |  OUT  | port_cap_limit | (  : 0.9000 ) |
|  out2 |  OUT  | port_cap_limit | (  : 0.9000 ) |
|  out3 |  OUT  | port_cap_limit | (  : 0.9000 ) |
|  out4 |  OUT  | port_cap_limit | (  : 0.9000 ) |
+-----------------------------------------------+
```

Related Commands

get_ports

report_ports

set_fanout_load

set_max_capacitance

set_max_transition

set_max_time_borrow

```
set_max_time_borrow
     borrow_value
     pin_or_instance_or_clock_list
```

Specifies the maximum time that can be borrowed by one stage from the next logic stage following a latch to meet timing constraints. Time borrowing is sometimes called cycle stealing. Time borrowing is performed automatically when performing timing analysis.

Use the `set_max_time_borrow` assertion on pins or waveforms. If the assertion is placed on a clock waveform, all latches triggered by this clock signal get the specified maximum time borrow limit. If both the clock waveform arriving at the clock pin of a latch and the clock pin itself have time borrow limit constraints, the maximum borrow limit on the latch is decided by the assertion on the latch clock pin.

Parameters

`borrow_value`
: Specifies the maximum amount of time that can be borrowed. The minimum value allowed is `0`, which disables time borrowing. The maximum allowed is equal to the pulse width minus the setup time: Max = pulse width - setup

`pin_or_instance_or_clock_list`
: Specifies the list of ports, instances or clock on which the borrow value is applied.The *`pin_or_instance_or_clock_list`* argument can be a collection.

Examples

```
set_max_time_borrow 2 [get_cells {i1, i2, i3}]
set_max_time_borrow 1.7 I4/g
set_max_time_borrow 2.5 [get_clocks CLK1]
```

set_max_transition

```
set_max_transition
     transition_value
     [-clock_path] | [-data_path]
     [-rise] | [-fall]
     object_list
```

Sets the maximum slew time limit (transition) on the specified ports of the top cell.

When set on a port, the slew time at the port must be less than or equal to the specified limit. Maximum transition limits on pins can come from library objects, which are the default maximum transition of the libraries or maximum transition on library cell pins. In this case, the most constraining limit, which is the smallest, will apply on pins and ports.

When you specify a clock waveform(s) as the object_list, you can use the additional parameters `-clock_path`, `-data_path`, `-rise`, and `-fall` to provide more precise specifications.

Use the `report_ports -type slew_limit` command to view the maximum transition limits set on ports of the top cell.

Parameters

`-clock_path`	Applies the max transition constraint only to pins in the clock network of the clocks defined in the *`object_list`*.
`-data_path`	Applies the max transition constraint only to data path pins, where the path is launched by the clocks defined in the *`object_list`*.
`-fall`	Applies the max transition constraint only to falling transitions.
`object_list`	Specifies a list or collection of ports, designs, or clocks.
`-rise`	Applies the max transition constraint only to rising transitions.
`transition_value`	Specifies the value of the maximum slew time limit (maximum transition), which must be greater than or equal to zero. The maximum slew time limit is in time units in the current session.

Examples

- The following command sets a maximum transition limit of `0.9` units on all the ports of the top cell:

  ```
  set_max_transition 0.9 [get_ports *]
  ```

- The following example shows a report of the maximum transition limit set on ports using the `report_ports` command, as shown in Example 30-7:

  ```
  set_max_transition 0.9 [get_ports *]
  report_ports -type slew_limit -pin [get_ports *]
  ```

Example 30-7 Maximum Transition Limit Set on Ports Report

Pin Name	Dir	Assertion	Value
in1	IN	slew_limit	(: 0.9000)
in2	IN	slew_limit	(: 0.9000)
out1	OUT	slew_limit	(: 0.9000)
out2	OUT	slew_limit	(: 0.9000)
out3	OUT	slew_limit	(: 0.9000)
out4	OUT	slew_limit	(: 0.9000)

- The following command sets the maximum transition limit of 0.001 units on pins in the `all_clocks` collection:

  ```
  set_max_transition 0.001 -clock_path [all_clocks]
  ```

Related commands:

`get_ports`

`report_ports`

`set_max_capacitance`

set_min_delay

```
set_min_delay
     [{-from |-rise_from | -fall_from} pin_or_clock_list]
     [{-through | -rise_through | -fall_through} pin_or_clock_list]
     [{-to | -rise_to | -fall_to} pin_or_clock_list]
     [-rise]
     [-fall]
     value
```

Specifies a minimum delay for a timing path. The `set_min_delay` command takes precedence over other path exceptions specified for the same path, except for `set_false_path` command specifications.

Use either the `-from`, `-to`, or `-through` option with the `set_min_delay` command to identify the paths. Specifying both the `-from` and `-to` options identifies a path with specific starting and ending points. If a clock is specified as the `-to` object, all primary outputs and registers clocked by that clock are used as an endpoint.

When you specify a sequential cell name for starting point of the timing path, the valid path starts from the clk pin. This means the latency and delay from clk to output Q is also included in the delay. For path segmentation, you can specify the path to start from Q output of the register by adding /Q after the cell name. In this case, path from clk to Q is not considered in the calculation. For example, in the following set of commands, the valid path starts from different points:

```
set_min_delay 0.66 -from TR1/BR1 -to BLK/BR1/D
set_min_delay 0.66 -from TR1/BR1/Q -to BLK/BR1/D
```

For the first command, the valid path starts from clk, and for the second command the valid path starts from the Q output. In the second case, the path from clk to Q is not considered. For an example of the results generated for paths with and without path segmentation, see Examples on page 1489.

Use only one `-from`, `-rise_from`, or `-fall_from`, `-to`, `-rise_to`, or `-fall_to` parameter per command. You can use multiple `-through` parameters in the same command.

Parameters

`-fall`	Applies the assertion at the falling edge of the signal on the path endpoints.
`-from |-rise_from | -fall_from` *`pin_or_clock_list`*	

Specifies a list of timing paths start points. The *pin_or_clock_list* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *pin_or_clock_list* argument can be a collection.

The valid start points are:

- Input port
- Bidirectional port
- Clock pin of sequential cell

  Clock pin of sequential cell is one of the following:

  - Structural clock pin that is part of latch or flip-flop.
  - Inferred clock pin in cells like macro cells. A pin is an inferred clock pin if it is associated with one of the following:
    - Reference end of setup check
    - Reference end of hold check
    - Reference end of removal check
    - Reference end of recovery check
    - Minimum period check
    - Minimum pulse width check
- Pin associated with set_input_delay

Specify one of the following:

`-from`
: Applies the minimum delay constraint to all paths originating from the *pin_or_clock_list* list. If a clock is specified as the `-from` object, all primary inputs and registers clocked by that clock are used as start points.

  By default, the `-from` option applies the constraints to both the rising and the falling edges.

`-rise_from` Applies the minimum delay constraint at the rising edge of the signal on the from pins.

`-fall_from` Applies the minimum delay constraint at the falling edge of the signal on the from pins.

`-rise` Applies the assertion at the rising edge of the signal on the path endpoints.

`-through | -rise_through | -fall_through` *`pin_or_clock_list`*

Specifies a list of timing paths through points.

The *`pin_or_clock_list`* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *`pin_or_clock_list`* argument can be a collection.

Specify one of the following:

`-through` When you use this parameter without using the `-from` or `-to` options, all paths that go through the specified pins have the specified delay. When you use it with both the `-from` and `-to` options, any path starting at any pin on the `-from` list *and* going through any point on the `-through` pin list *and* going to any point on the `-to` list is affected by the constraint.

By default, the `-through` option applies the constraint to both the rising and the falling edges.

`-rise_through` Applies the assertion at the rising edge of the signal on the through pins.

`-fall_through` Applies the assertion at the falling edge of the signal on the through pins.

`-to | -rise_to | -fall_to` *`pin_or_clock_list`*

Specifies a list of timing paths end points.

The *pin_or_clock_list* argument can contain either object IDs or hierarchical names relative to the current module or clock names. The pins can be on intermediate hierarchical boundaries. The *pin_or_clock_list* argument can be a collection.

The valid end points are:

- Output port
- Bidirectional port
- Data pin of sequential cell

  Data pin of sequential cell is one of the following:

  - Structural data pin that is part of latch or flip-flop.
  - Inferred data pin in cells like macro cells. A pin is an inferred data pin if it is associated with one of the following:
    - Signal end of setup check
    - Signal end of hold check
    - Signal end of non-sequential setup check
    - Signal end of non-sequential hold check
    - Signal end of removal check
    - Signal end of recovery check
- Pin associated with `set_external_delay`
- Data pin of clock gating cell

Select from one of the following:

`-to`
: Applies the minimum delay constraint to all paths ending in the `-to` pin list. If a clock is specified as the `-to` object, all primary outputs and registers clocked by that clock are used as endpoint.

  By default, the `-to` option applies the constraint to both the rising and the falling edges.

	`-rise_to`	Applies the minimum delay constraint at the rising edge of the signal on the to pins.
	`-fall_to`	Applies the minimum delay constraint at the falling edge of the signal on the to pins.
value	Specifies the minimum delay value.	

Examples

- The following command specifies a minimum delay of `2.5` for paths from `in1` to `out1`:

```
set_min_delay 2.5 -from in1 -to out1
```

- The following commands show the difference in `report_timing` results when you specify a minimum delay with and without path segmentation:

```
set_min_delay 9 -from dff1 -to dff2/d
set_min_delay 9 -from dff1/q -to dff2/d
```

  The `report_timing` results for the first command with path segmentation are:

```
report_timing -early -from dff1
Path 1: VIOLATED Hold Check with Pin dff2/cp
Endpoint:   dff2/d (v) checked with  leading edge of 'clk'
Beginpoint: dff1/q (^) triggered by  leading edge of 'clk'
Other End Arrival Time          0.000
+ Hold                        -1.956
+ Path Delay                   9.000
= Required Time                7.044
  Arrival Time                 6.148
  Slack Time                  -0.896
     Clock Rise Edge                     0.000
     = Beginpoint Arrival Time           0.000
     +-------------------------------------------------------------+
     | Instance |     Arc     |  Cell  | Delay | Arrival | Required |
     |          |             |        |       |  Time   |   Time   |
     |----------+-------------+--------+-------+---------+----------|
     |          | clk ^       |        |       |   0.000 |    0.896 |
     | dff1     | cp ^ -> q ^ | dfn3d1 | 4.250 |   4.250 |    5.146 |
     | inv3     | i ^ -> zn v | in01d1 | 1.898 |   6.148 |    7.044 |
     | dff2     | d v         | dfn3d1 | 0.000 |   6.148 |    7.044 |
     +-------------------------------------------------------------+
```

  The `report_timing` results for the second command without path segmentation are:

```
report_timing -early -from dff1
```

```
Path 1: MET Hold Check with Pin dff2/cp
Endpoint:   dff2/d (v) checked with  leading edge of 'clk'
Beginpoint: dff1/q (^) triggered by  leading edge of 'clk'
Other End Arrival Time          0.000
+ Hold                         -1.956
+ Phase Shift                   0.000
= Required Time                -1.956
  Arrival Time                  6.148
  Slack Time                    8.104
     Clock Rise Edge                      0.000
     = Beginpoint Arrival Time            0.000
     +-------------------------------------------------------------+
     | Instance |     Arc     |  Cell  | Delay | Arrival | Required |
     |          |             |        |       |  Time   |   Time   |
     |----------+-------------+--------+-------+---------+----------|
     |          | clk ^       |        |       |   0.000 |   -8.104 |
     | dff1     | cp ^ -> q ^ | dfn3d1 | 4.250 |   4.250 |   -3.854 |
     | inv3     | i ^ -> zn v | in01d1 | 1.898 |   6.148 |   -1.956 |
     | dff2     | d v         | dfn3d1 | 0.000 |   6.148 |   -1.956 |
     +-------------------------------------------------------------+
```

Related Commands

set_input_delay

set_output_delay

set_false_path

set_max_delay

set_multicycle_path

set_min_pulse_width

```
set_min_pulse_width
     [-low] [-high]
     value list
```

Selects the most restrictive value when multiple minimum pulse width checks are set.

Note: The `timing_disable_lib_pulsewidth_checks` global has no effect on pulse width checks created by the `set_min_pulse_width` command.

Parameters

`-high`	Applies the minimum pulse width check to high signal levels only. If neither the `-low` or `-high` option is specified, the value is applied to both low and high signal levels.
`list`	Specifies a list of pins or clock waveforms to which the minimum pulse width check is to be applied. If the list is not specified, the minimum pulse width check applies to all the pins in the clock tree of the current design. If a clock waveform is specified, all the pins driven by the clock are affected.
`-low`	Applies the minimum pulse width check to low signal levels only. If neither the `-low` or `-high` option is specified, the value is applied to both low and high signal levels.
`value`	Specifies a non-negative floating minimum pulse width check value. No default value is assumed for minimum pulse width checks.

Examples

- The following command sets a minimum pulse width, both low and high, check of `1.0` to the clock network of the current design:

  ```
  set_min_pulse_width 1.0
  ```

- The following command sets a minimum pulse width low check of `2.0` to all the pins clocked by the `CLK1` clock:

  ```
  set_min_pulse_width -low 2.0 [get_clocks CLK1]
  ```

- The following command sets a minimum pulse width `-high` check of `1.0` to the `CP` clock pin of register `ff1`, as shown in Example 30-8:

Example 30-8 Minimum Pulse Width -high Report

```
set_min_pulse_width -high 1.0 ff1/CP
report_timing -check_clocks -rise_to ff1/CP
Path 1: VIOLATED User PulseWidth Check with Pin ff1/CP
       Endpoint:   ff1/CP (^) checked with  leading edge of 'CLK'
       Beginpoint: clk    (v) triggered by trailing edge of 'CLK'
       Path Groups:  {CLK}
       Other End Arrival Time      0.10
       - PulseWidth                1.00
       + Phase Shift               0.70
       = Required Time            -0.20
       - Arrival Time              0.45
       = Slack Time               -0.65
            Clock Fall Edge                  0.35
            = Beginpoint Arrival Time        0.35
            +-------------------------------------------------------------+
            | Instance |    Arc     |  Cell | Delay | Arrival | Required |
            |          |            |       |       |  Time   |   Time   |
            |----------+------------+-------+-------+---------+----------|
            |          | clk v      |       |       |    0.35 |    -0.30 |
            | inv1     | A v -> Z ^ | IV    |  0.10 |    0.45 |    -0.20 |
            | ff1      | CP ^       | FD1   |  0.00 |    0.45 |    -0.20 |
            +-------------------------------------------------------------+
```

Related Commands

report_analysis_coverage

report_min_pulse_width

report_timing

set_mode

```
set_mode
     [-type cell]
     list_of_modes
     instance_list
```

Specifies the Liberty timing library modes to be active for a specific instance.

A library mode is a subset of active timing arcs that is defined in the timing library. Library modes are similar to conditional timing arcs, but do not require the application of constants to set. Instead of changing a cell's mode by applying constants to its input pins, you can use the `set_mode` command to refer to sets of active arcs by name (such as `READ` or `WRITE`, for a memory cell).

By default, all library modes in the timing library are active. If you specify one or more library modes using the `set_mode` command, only the timing arcs consistent with the specified mode(s) are active for the instance. All other library modes become inactive.

To reset the library modes for an instance, use the `reset_mode` command.

Parameters

instance_list	Specifies the instances to which to apply the library mode.
list_of_modes	Specifies the name of the library mode value defined in the timing library. You can specify one or more mode values, but only one mode value per defined mode group in the Liberty library.
`-type cell`	Sets the library mode at the instance level. **Note:** Setting the library mode at the instance level is the only form of `set_mode` that is currently supported. This parameter exists to support constraints coming from external sources.

Examples

- The following command specifies that only the timing arcs consistent with the `tst_mode_disable` library mode are active for instances of `i_ram`. The `tst_mode_enable` library mode is made inactive.

  ```
  set_mode tst_mode_disable [get_cells i_ram]
  ```

- The following command reports the status of all library modes associated with `i_ram`. It shows that `tst_mode_disable` is active, and `tst_mode_enable` is inactive.

```
report_mode i_ram

+-------------------------------------------------+
|         Mode Group testmode of i_ram            |
|-------------------------------------------------|
|    Mode Name      |  Status   |  Condition      |
|-------------------+-----------+-----------------|
|  tst_mode_enable  | Inactive  |  TST_BYPASS     |
| tst_mode_disable  | ACTIVE    |  !(TST_BYPASS)  |
+-------------------------------------------------+
```

- If you specify the `report_inactive_arcs` command, the report shows the disabled arcs for the `i_ram` instances are from the `tst_mode_enable` library mode:

```
+-------------------------------------------------------------------------------------------------+
| Flags :       | const               | propagated constant     |               |                 |
|               | snipped             | loop snipped arcs       |               |                 |
|               | disable             | set_disable_timing      |               |                 |
|               | disable_clock_gating| set_disable_clock_gating|               |                 |
|               | library_disable     | set_disable_cell_timing |               |                 |
|               | Missing_Phase       | No Arc Phase Data       |               |                 |
| From          | To                  | DisableType             | ArcType       | Reason          |
|---------------+---------------------+-------------------------+---------------+-----------------|
| i_ram/D[31] ^ | i_ram/Q[31] ^       | Library_Mode            | combinational | mode testmode = |
|               |                     |                         |               | tst_mode_ebale  |
| i_ram/D[31] v | i_ram/Q[31] v       | Library_Mode            | combinational | mode testmode = |
|               |                     |                         |               | tst_mode_ebale  |
| i_ram/D[30] ^ | i_ram/D[30] ^       | Library_Mode            | combinational | mode testmode = |
|               |                     |                         |               | tst_mode_ebale  |
| i_ram/D[30] v | i_ram/Q[30] v       | Library_Mode            | combinational | mode testmode = |
|               |                     |                         |               | tst_mode_ebale  |
+-------------------------------------------------------------------------------------------------+
```

set_multicycle_path

```
set_multicycle_path
     [-hold | -setup]
     [-start | -end]
     [{-from | -rise_from | -fall_from} pin_or_clock_list]
     [{-through | -rise_through | -fall_through} pin_list]
     [{-to | -rise_to | -fall_to} pin_or_clock_list]
     [-rise]
     [-fall]
     number_of_cycles
```

Specifies multicycle paths between specific timing paths in the design or between clock domains. Multicycle paths are timing paths that require more than one clock period for execution. By default, all paths are considered as one cycle paths for setup and hold check evaluation. The `set_multicycle_path` command lets you specify additional cycles for the paths.

Note: Only internally generated scripts such as the `write_assertions` command uses `-rise` and `-fall` as options. Use the `{-rise_from | -fall_from}` and the `{rise_to | -fall_to}` options.

Parameters

`-fall` — Applies the assertion at the falling edge of the signal on the path endpoints.

`{-from | -rise_from | -fall_from}` *`pin_or_clock_list`*

	Specifies the pins or clocks that are at the start of the paths. If the `-from` option is used without the `-to` option, all paths originating from the *`pin_or_clock_list`* are multicycle paths. Use either the `-from`, `-to`, or `-through` option with the `set_multicycle_path` command to identify the paths. Specifying both the `-from` and `-to` options identifies a path with specific starting and ending points. The *`pin_or_clock_list`* variable can contain either object IDs, hierarchical names relative to the current module, or clock names. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified as the object for the `-from*` options. The *`pin_or_clock_list`* variable can be a collection. Use only one `-from`, `-rise_from`, or `-fall_from` option per command.
`-hold | -setup`	Specifies whether the multicycle applies to the `-hold` or `-setup` times. The `-setup` parameter applies multicycles for setup and effects the hold check. It does not change the hold multiplier. The `-hold` parameter applies multicycles for hold times only. *Default:* If you do not use `-hold` or `-setup` parameters, the software applies multicycles to setup times and effects the hold times.
`number_of_cycles`	Specifies the number of cycles allowed for the path.
`-rise`	Applies the assertion at the rising edge of the signal on the path endpoints.
`-start | -end`	Specifies whether the multicycle adjustment is to be applied relative to the starting clock edge for the path or relative to the ending clock edge for the path. Use these options for multiple clocks designs. By default, the setup check is relative to the end clock and the hold check is relative to the start clock.

`{-through | -rise_through | fall_through}` *`pin_list`*

Specifies the pins that the path goes through. When used without the `-from` or `-to` option, all paths that go through the `-through` pins are multicycle paths.

When used with both the `-from` and `-to` options, any path starting at any pin on the `-from` list and going through any point on the `-through` pin list and going to any point on the `-to` list is affected by the assertion.

The *`pin_list`* can contain either object IDs or hierarchical names relative to the current module, or it can be a collection. The pins can be on intermediate hierarchical boundaries.

Use only one `-from` and one `-to` option, but you can use many `-through` options in the same command.

By default, the `-through` option applies the assertion to both the rising and the falling edges. The `-through` *`pin_list`* can be a list of pin names, ID, or a collection of pins.

Using the `-rise_through` option applies the assertion at the rising edge of the signal on the through pins.

Using the `-fall_through` option applies the assertion at the falling edge of the signal on the through pins.

`{-to | -rise_to | -fall_to}` *`pin_or_clock_list`*

Specifies the pins that are at the end of the paths. If the `-to` option is used without the `-from` option, all paths ending in the `-to` pin list are multicycle paths.

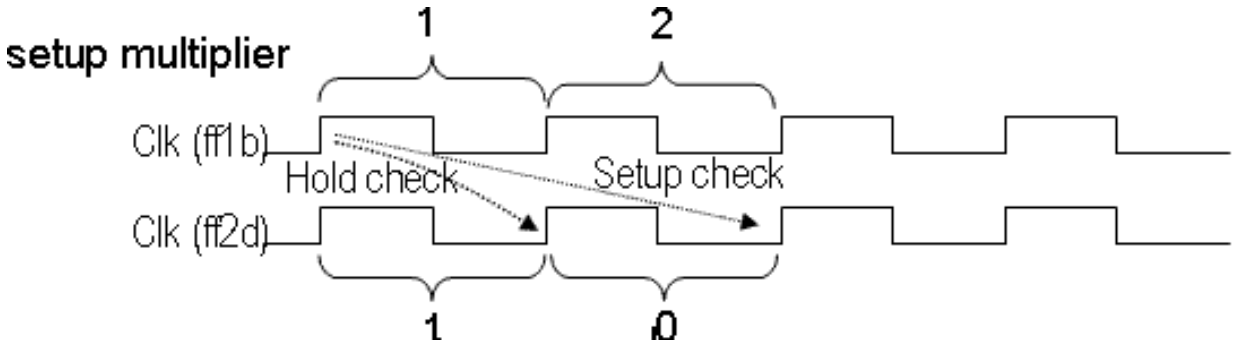
The *`pin_or_clock_list`* argument can contain either object IDs or hierarchical names relative to the current module, or clock names. The pins can be on intermediate hierarchical boundaries. Leaf design cells (instances) can also be specified as the object for the `-to*` options. The *`pin_or_clock_list`* argument can be a collection. Use only one `-to` option per command.

Use only one `-to`, `-rise_to`, or `-fall_to` option per command.

Examples

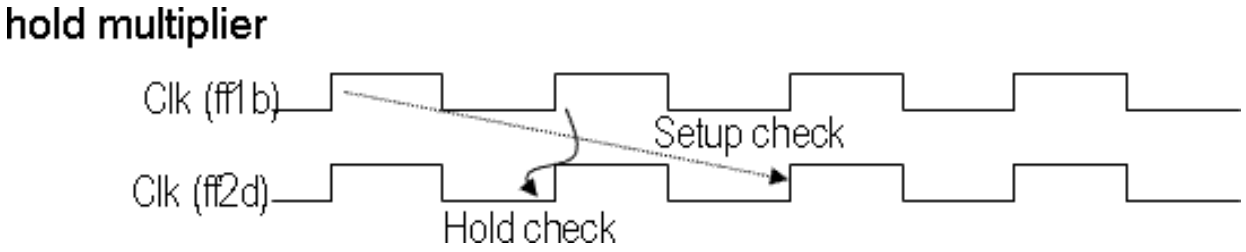
- The following command sets all paths between ff1b + ff2d to two cycles for setup:

```
set_multicycle_path -setup 2 -from {ff1b} -to {ff2d}
```



- The following command moves hold check to start clock preceding edge:

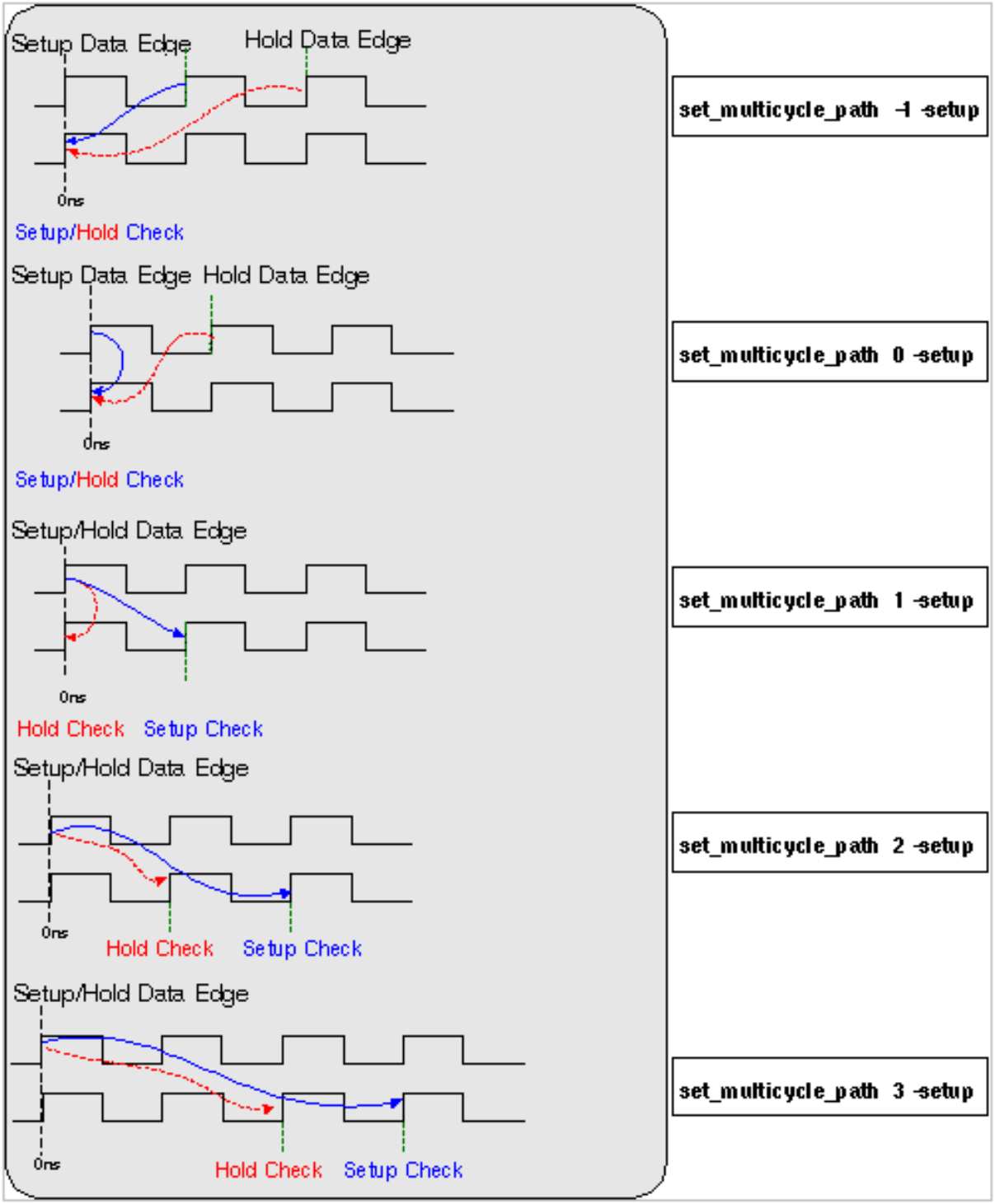
```
set_multicycle_path -hold 1 -from {ff1b} -to {ff2d}
```



- The following commands show setup multiplier values from -1 to 3:

```
set_multicycle_path -1 -setup
set_multicycle_path 0 -setup
set_multicycle_path 1 -setup
set_multicycle_path 2 -setup
```

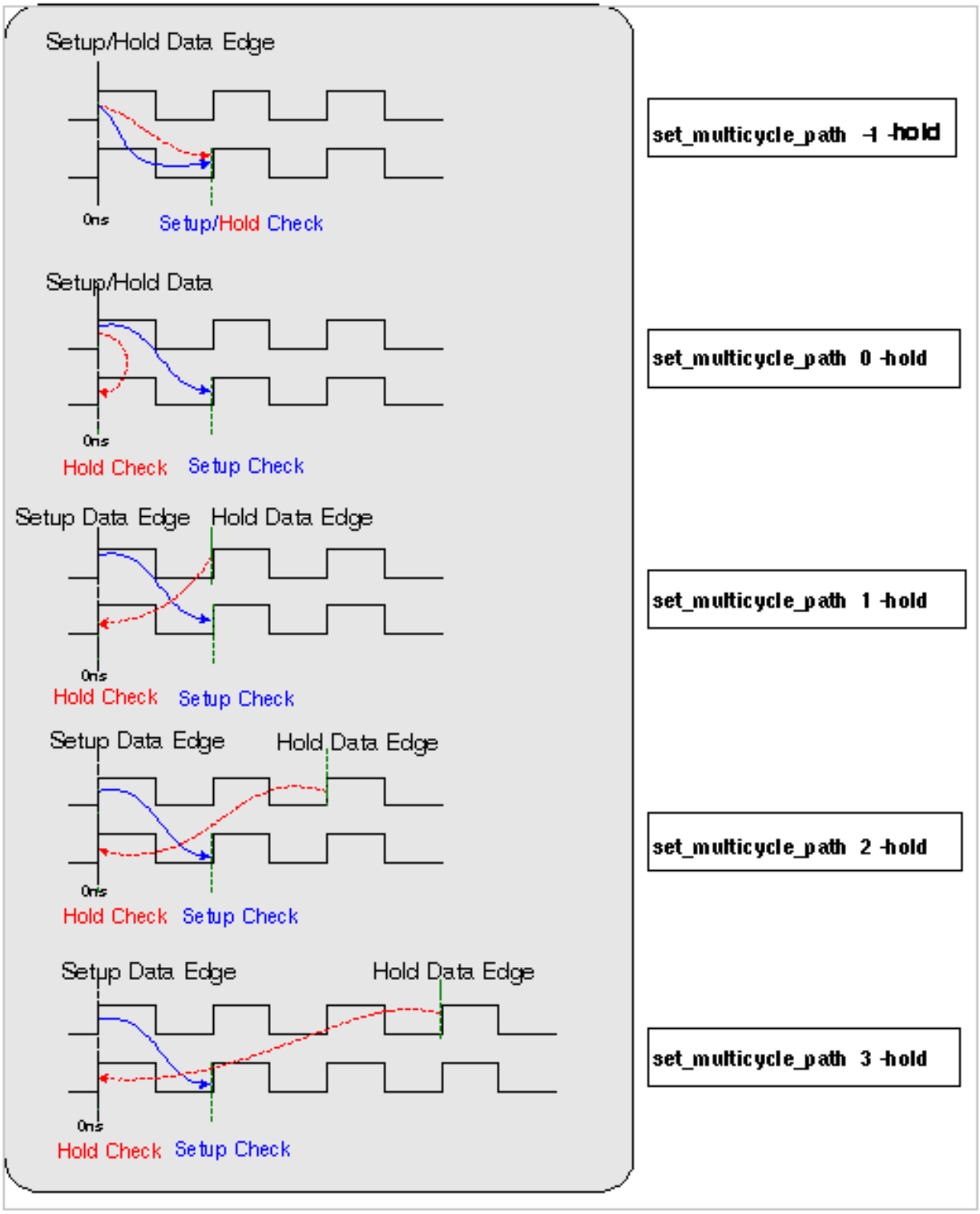
```
set_multicycle_path 3 -setup
```



- The following commands show setup multiplier values from -1 to 3:

```
set_multicycle_path -1 -hold
set_multicycle_path 0 -hold
set_multicycle_path 1 -hold
set_multicycle_path 2 -hold
```

```
set_multicycle_path 3 -hold
```



Related Commands

set_false_path

set_max_delay

set_min_delay

set_output_delay

```
set_output_delay
     [-clock clock_name]
     [-clock_fall]
     [-rise] [-fall]
     [-min] [-max]
     delay_value
     [-add_delay]
     [-network_latency_included] [-source_latency_included]
     [-group_path group_name]
     [-reference_pin pin_name]
     [-level_sensitive]
     port_or_pin_list
```

Specifies the data required time at the output port and on hierarchical ports.

Parameters

`-add_delay`	Adds delay information to the existing output delay. The additional delay information can be for multiple paths relative to different clocks or clock edges leading to an input port or internal input pins. If you do not use the `-add_delay` option, the software uses the last `set_output_delay` related to a pin when there are multiple `set_output_delay` statements for the same pin.
`-clock` *`clock_name`*	Specifies that the delay is relative to the specified clock.
`-clock_fall`	Specifies that the delays are with respect to the falling edge of the clock waveform. If you do not specify the `-clock_fall` option, the delay is with respect to the rising edge of the clock waveform.
`delay_value`	Specifies the value for the path delay.
`-fall`	Specifies that the output delay refers to the falling edge of the signal at specified ports or pins. If you do not specify the `-fall` option, the output delay applies to both the edges.
`-group_path` *`group_name`*	Specifies the group name where paths ending at the specified ports or pins are to be added.

`-level_sensitive` Models output delay that is captured by a positive level sensitive latch. For setup checks, the timing system checks the arrival time at the output port with the opening edge of the latch. The timing system treats any arrival time exceeding the opening edge as negative slack. For hold checks, the timing system performs the checks with respect to the closing edge of the latch.

Use the `-clock` parameter to model a positive level sensitive latch, or the `-clock_fall` parameter to model a negative level sensitive latch.

`-max` Specifies that the delay refers to the setup analysis. If you do not specify the `-max` parameter, the output delay applies to both setup and hold analysis.

`-min` Specifies that the delay refers to the hold analysis. If you do not specify the `-min` parameter, the output delay applies to both setup and hold analysis.

`-network_latency_included`

Specifies that the clock network latency should not be added to the output delay value. If you do not specify this option, the clock network latency of the related clock is added to the output delay value. This parameter has no effect if the specified clock is in propagated mode.

`port_or_pin_list` Specifies a list of output port or pin names. The *`port_or_pin_list`* argument can be a collection.

`-reference_pin` *`pin_name`*

Adds source and network latency of the specified reference pin to the output delay value in propagated mode. A reference pin is a leaf pin in the fanin cone of the clock source of the clock that you specify using the `-clock` parameter. You cannot use the `-reference_pin` parameter with the `-network_latency_included` or `-source_latency_included` parameters.

`-rise` Specifies that output delay refers to the rising edge of the signal at the specified ports or pins. If you do not specify the `-rise` option, the output delay applies to the both edges.

`-source_latency_included`

Specifies that the clock source latency should not be added to the output delay value.

Examples

- The following example sets the input and output delays for the bidirectional `INOUT2` port.The input signal arrives at `INOUT2` at `4.5` units after the falling edge of `CLKB`. The output signal is required at `INOUT2` at `2.5` units before the rising edge of `CLKD`:

  ```
  set_input_delay 4.5 -clock CLKB -clock_fall { INOUT2 }
  set_output_delay 2.5 -clock CLKD { INOUT1 }
  ```

- The following example shows how to use the `-group_path` option to add ports into a named group:

  ```
  set_output_delay 2.5 -max -clock CLK -group_path busC {busC[*]}
  ```

Related Information

`all_outputs`

`create_clock`

`report_ports`

`set_input_delay`

`set_load`

set_propagated_clock

```
set_propagated_clock
     pin_or_clock_list
```

Puts the `propagated_clock` assertion on the specified pin, port, or clock object. This will cause all clock endpoints in the fanout of the specified object to receive propagated clock timing unless there is a `set_clock_latency` with higher precedence. If there are multiple `set_clock_latency` and `set_propagated_clock` assertions between the clock waveform, clock root and clock endpoint, the closest assertion to the clock endpoint takes precedence.

There are two methods of clock propagation: depending on whether the clock is ideal or propagated.

- When a clock is ideal, the delay from a clock port to a register clock pin (network insertion delay) comes from design constraints using the `set_clock_latency` command.
- When a clock is propagated, the network insertion delay is computed from the actual gates and interconnects in the clock network.

To set the propagated_clock property on all clock waveform objects, use the following command:

```
set_propagated_clock [all_clocks]
```

This command must be the last line in the SDC file to affect all clocks. Whether all clock endpoints receive propagated or ideal timing depends on the precedence of `set_clock_latency` in the fanout.

The `set_propagated_clock [all_clocks]` command puts the `propagated_clock` assertion on all clocks except virtual clocks. When you put the assertion on a real clock that is used to define I/O timing (with `set_input_delay -clock` or `set_output_delay -clock`), the software does not take the specified network latency into account. Instead, the software calculates the network latency, which changes the I/O timing. You can change this behavior by using the following global variable:

```
set_global timing_io_use_clock_network_latency always
```

You can use this command after creating the clock with the `create_clock` command.

Parameters

pin_or_clock_list	Specifies the objects on which you want to put the propagated_clock assertion. The *pin_or_clock_list* argument can be a collection. When a pin is specified, it affects the propagation mode for all the registers in the transitive fanout (TFO) of the pin.

Examples

- The following command puts the propagated_clock assertion on the clock waveform CLK1:

  ```
  set_propagated_clock CLK1
  ```

- The following command puts the propagated_clock assertion on all clock waveforms:

  ```
  set_propagated_clock [all_clocks]
  ```

Related Information

all_clocks

get_propagated_clock

report_clocks

set_clock_latency

set_clock_uncertainty

set_resistance

```
set_resistance
     [-max] [-min]
     resistance_value
     net_list
```

Sets the wire resistance value on the specified nets of the top cell, which is used to compute the net delay of the specified nets. The resistance that you specify using this command overrides the resistance from wire load models and SPEF.

Note: In the best case-worst case analysis mode, the minimum resistance is used to compute the early path delay and the late path delay for the minimum corner. The maximum resistance is used to compute the early path delay and the late path delay for the maximum corner.

Parameters

`-max`	Specifies that the asserted resistance is the maximum resistance. *Default:* If neither the `-min` option nor the `-max` option is specified, the resistance value is used as the minimum and maximum resistance.
`-min`	Specifies that the asserted resistance is the minimum resistance. Default: If neither the `-min` option nor the `-max` option is specified, the resistance value is used as the minimum and maximum resistance.
`net_list`	Specifies a list of nets on which the wire resistance is set.
`resistance_value`	Specifies the value for the wire resistance. This value is in units found in the target technology library.

Examples

- The following example set the maximum wire resistance of `2.0` units on the net `n1`:

  ```
  set_resistance -max 2.0 [get_nets n1]
  ```

- The following example set the minimum wire resistance of `1.2` units and a maximum wire resistance of `2.0` units on the net `n1`:

  ```
  set_resistance 1.2 -min [get_nets n1]
  ```

```
set_resistance 2.0 -max [get_nets n1]
```

Related Information

report_net

report_timing

set_drive

set_load

31

Signal Integrity Commands

fixACLimitViolation

```
fixACLimitViolation
     [-excNetFile filename]
     [-selNetFile filename]
     [-useRuleFile filename]
```

Repairs violations associated with AC current density, wire-self-heat, Joules heat, or signal wire electromigration on signal wires by adding a buffer to split up the net load to any net marked as a violation with the `verifyACLimit` command. Use this command in conjunction with the `verifyACLimit` command. The `verifyACLimit` command marks the violating net for the `fixACLimitViolation` command to repair. The `verifyACLimit` and `fixACLimitViolation` commands typically are used in the following command order:

```
setAnalysisMode -checkType hold
setExtractRCMode -engine detail
extractRC
verifyACLimit
fixACLimitViolation
runQX -rcType decoupledRC -spefOutput StarN
```

Note: You must perform AC current density analysis before repairing AC current limit violations. AC current limit violation repairs do not modify or repair clock nets. You must set the extraction mode to detailed and use the `-noReduce` option with the `setExtractRCMode` command before using this command.

Parameters

`-excNetfile` *`fileName`*

Specifies a file that contains the names of nets to exclude from AC limit violation repair.

Default: Repairs AC limit violations on all nets marked as violations

`-selNetFile` *`fileName`*

Specifies a file that contains the names of nets for which to perform AC limit violation repair.

Default: Repairs AC limit violations on all nets marked as violations

`-useRuleFile` *`fileName`*

Specifies a rule file that contains net names and the corresponding nondefault rules that will repair the AC limit violations for the nets. Any nets with violations listed in *`fileName`* will have their routing rule changed to the corresponding nondefault rule in the file and all routed wiring of the net is removed. Note that the net will not be re-routed. See below for an example of how one could also re-route the wiring. Any other nets with violations will use buffer insertion to reduce the capacitance on the net in order to fix the violation.

Example

- An example of using a rule file to repair the violations and reroute the wiring:

```
setExtractRCMode -engine detail
verifyACLimit -ruleFile aclimit.rule -report sean.ac.log.old -detailed
-scaleIrms 0.8
fixACLimitViolation -useRuleFile aclimit.rule

setAttribute -net <all vio nets> with non-default rule
selectNet <all vio nets>
setNanoRouteMode -routeSelectedNet true
ecoRoute

fixACLimitViolation
```

fixGlitchViolation

```
fixGlitchViolation
     [-ignoreCriticalNets]
     [-viofile filename]
```

Important

Command `fixGlitchViolation` is obsolete. However, it will remain functional as an internal sub routine to support the internal operations of the `optDesign` command.

Repairs glitch violations based on the peak noise. The `optDesign -postRoute -si` command automatically calls this command; therefore you do not need to call it explicitly.

Note: The Common Timing Engine (CTE) is not supported with this command.

Parameters

`-ignoreCriticalNets`	Ignores nets marked as critical in the database. To check if a net is marked as critical in the database, use the `dbIsNetCritical` *`net_name`* database command.
`-viofile` *`filename`*	Reads violating nets.

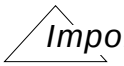
Example

- The following commands repair glitch violations while ignoring timing critical nets:

```
runCeltIC
fixGlitchViolation -ignoreCriticalNets
```

fixNoiseDelay

```
fixNoiseDelay
     [-slack value]
```



Command `fixNoiseDelay` is obsolete. However, it will remain functional as an internal sub routine to support the internal operations of the `optDesign` command.

Repairs the delta delay caused by noise. The `fixNoiseDelay` command takes timing into consideration. Use this command after detailed RC extraction.

Note: CTE is not supported with this command.

Parameters

`-slack value`	Repairs nets with slack less than or equal to the specified *value*, in nanoseconds. Nets are repaired during in-place optimization (IPO).

Example

- The following command repairs the delta delay caused by noise:

  ```
  fixNoiseDelay
  ```

  You can use the `fixNoiseDelay` command in sequence with the following commands:

  ```
  runCeltIC
  fixNoiseDelay -slack 0
  ```

getCdbFileWithAnalysisMode

```
getCdbFileWithAnalysisMode [-max | -min]
```

Returns the cdB files used for setup (`-max`) or hold (`-min`) analysis mode. If you do not specify `-max` or `-min`, returns the cdB files for the currently specified analysis mode setting.

Parameters

`-max`	Returns the cdB files used in setup analysis mode.
`-min`	Returns the cdB files used in hold analysis mode.

getEchoFileWithAnalysisMode

```
getEchoFileWithAnalysisMode [-max | -min]
```

Returns the ECHO model files used for setup (`-max`) or hold (`-min`) analysis mode. If you do not specify `-max` or `-min`, returns the ECHO model files for the currently specified analysis mode setting.

Parameters

`-max`	Returns the ECHO files used in setup analysis mode.
`-min`	Returns the ECHO files used in hold analysis mode.

getSIMode

```
getSIMode
     [-acceptableWNS]
     [-analyzeNoiseThreshold]
     [-deltaDelayThreshold]
     [-detailedReports]
     [-effortLevel]
     [-extractionEngine]
     [-fixDelay]
     [-fixDRC]
     [-fixGlitch]
     [-fixHoldIncludeXtalkSetup]
     [-insCeltICPostTcl]
     [-insCeltICPreTcl]
     [-irDropFile]
     [-maxNrIter]
     [-noiseProcess]
     [-noiseTwfMode]
     [-numQXcpu]
     [-outputPath]
     [-quiet]
     [-runStandAloneNanoRoute]
     [-runTrimMetalFill]
     [-targetPathGroups]
     [-saveSIFixDB]
     [-usePrevCeltICRunDir]
     [-usePrevSpefFile]
     [-usePrevTwfFile]
     [-usePrevWdbName]
     [-wrouteExtraConfig]
```

Displays the following information about a setSIMode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setSIMode` parameters.

Parameters

parameter_names	Displays information for the specified parameters. You can specify one or more parameters. See setSIMode for descriptions of the parameters you can specify.
-quiet	Displays the current settings for the specified parameters in Tcl list format only. If you specify -quiet without any parameters, the software displays the current settings of all setSIMode parameters in Tcl list format.

Examples

- The following command displays the current setting for the -runTrimMetalFill parameter:

  ```
  getSIMode -runTrimMetalFill
  ```

  The software displays the following information:

  ```
  -runTrimMetalFill false              # bool, default=false
  false
  ```

- The following command displays the current setting for the -runTrimMetalFill parameter in Tcl list format only:

  ```
  getSIMode -runTrimMetalFill -quiet
  ```

  The software displays the following information:

  ```
  false
  ```

- The following command displays the current setting of the -analyzeNoiseThreshold, -deltaDelayThreshold, and -noiseProcess parameters:

  ```
  getSIMode -analyzeNoiseThreshold -deltaDelayThreshold -noiseProcess
  ```

  The software displays the following information:

  ```
  -analyzeNoiseThreshold 20          # int, default=20
  -deltaDelayThreshold -1            # float, default=-1
  -noiseProcess 90nm                # enums={130nm 180nm 90nm 65nm}, default=90nm
  {analyzeNoiseThreshold 20} {deltaDelayThreshold -1} {noiseProcess 90nm}
  ```

- The following command displays the current settings for all `setSIMode` parameters in Tcl list format only:

  ```
  getSIMode -quiet
  ```

  The software displays the following information:

  ```
  {acceptableWNS same} {analyzeNoiseThreshold 20} {deltaDelayThreshold -1}
  {effortLevel high} {extractionEngine detail} {insCeltICPostTcl {}}
  {insCeltICPreTcl {}} {irDropFile {}} {maxNrIter 2} {noiseProcess 90nm}
  {noiseTwfMode {}} {numQXcpu 1} {outputPath ./result} {runStandAloneNanoRoute
  false} {runTrimMetalFill false} {targetPathGroups {reg2reg in2out in2reg
  reg2out}} {usePrevCeltICRunDir {}} {usePrevSpefFile {}} {usePrevTwfFile {}}
  {usePrevWdbName {}} {wrouteExtraConfig {}}
  ```

readTransitionFile

```
readTransitionFile
     -view viewName
     -file fileName
```

Reads external transition files (for MMMC or non-MMMC designs) that will be used to perform maximum transition violation fixing.

The content of the transition file must be in the following format:

```
setTranTime [-add] [-maxR slewValue | -maxF slewValue] { pinNames }
```

Where:

- `-add`: Specifies that the transition value is incremental.
- `-maxR | -maxF`: Specifies the maximum rise or fall transition.

  Note: Minimum transition values will be ignored while performing transition violation fixing.
- *`pinNames`*: Specifies the pin names for which the transition value is specified.

Parameters

`-file` *`fileName`*	Specifies the name of the file that contains information about the transition time.
`-view` *`viewName`*	Specifies the view name for which the transition file will be read in case of MMMC designs. Ensure that you specify the transition files for all setup views to perform transition violation fixing. *Important* This parameter is needed only when using MMMC designs.

Example

- The following command reads an external transition file called `sampleFile` for non-MMMC designs:

  ```
  readTransitionFile -file sampleFile
  ```

- The following commands read view specific external transition files for an MMMC design:

```
readTransitionFile -view view1 -file sampleFile1
readTransitionFile -view view2 -file sampleFile2
```

Related Topics

- SI Closure in the *Encounter Flat Implementation Flow Guide*
- Partition SI Closure in the *Encounter Hierarchical Implementation Flow Guide*
- Top-Level SI Closure in the *Encounter Hierarchical Implementation Flow Guide*

reportCouplingCaps

`reportCouplingCaps` *`net_names`*

Reports the coupling capacitance of the specified nets.

Note: Before using this command, either run the `spefIn` command, or set the extraction mode to `-engine detail` or `-engine CCE` with the `setExtractRCMode` command.

Parameters

`net_names`	Specifies the nets for which to report coupling capacitance.

Example

- The following commands report the coupling capacitance of nets:

```
setExtractRCMode -engine detail
extractRC
reportCouplingCaps top/block1/cellA/netB
```

runCeltIC

```
runCeltIC
     [-cdb {list_of_files}]
     [-echo {list_of_files}]
     [-effort {high | low}]
     [-genScriptOnly]
     [-irDrop {ir_drop_file}]
     [-infSW]
     [-initSW]
     [-insPreTcl {variables}]
     [-native]
     [-process {180nm | 130nm | 90nm | 65nm | 45nm}]
     [-saveCdb]
     [-saveEcho]
     [-saveXilm]
     [-spef filename]
     [-twf filename]
     [-update | -noUpdate]
     [-view]
```

Important

Command `runCeltIC` is obsolete. Use the `timeDesign -postRoute -si` command to perform SI analysis. The parameters of the `runCeltIC` command were either replaced by parameters of the `setSIMode`, `timeDesign`, and `saveModel` commands, or do not have any replacement.

Runs the SI analysis engine to perform crosstalk analysis.

Note: Starting in the SoC 7.1 release, Encounter calls native SI analysis functionality by default. Native SI analysis replaces standalone CeltIC NDC in all crosstalk analysis and repair operations. Standalone CeltIC NDC is no longer supported.

Parameters

`-cdb {list_of_files}`

Specifies a list of common noise library (cdB) files to use.

`-echo {list_of_files}`

Specifies a list of common noise library (ECHO) files to use.

`-effort {high | low}`

Specifies the level of effort the tool puts forth for a particular usage model. The `low` effort mode performs a much faster analysis, while trading minimal accuracy for improved run time. The high effort mode uses the best accuracy and should always be selected for final sign-off.

Default: `high`.

Note: If you specify `high`, ensure that you load cdB libraries that were created with the most current version of the `make_cdb` utility. Often, new features do not function properly when used with an old cdB library.

`-genScriptOnly` Writes the script (`run_celtic.tcl`) necessary to run SI analysis.

`-infSW` Performs analysis using infinite switching windows. This allows all aggressors to switch at the same time. Use this option when the timing constraints are not well defined.

`-initSW` Resets and initializes the switching windows.

Default: Uses the existing switching windows in memory.

`-insPreTcl {`*`variables`*`}`

Changes the default SI analysis environment variable values to the specified values. Use the following format to specify the variable values:

```
{set_parm variable1 value ; \
 set_parm variable2 value ; \
 set_parm variable3 value ;}
```

Alternatively, you can specify a tcl file that contains a list of variables. For example,

`-insCeltICPreTcl {source `*`directory_path/file_name`*`.tcl}`

If you specify a file, you must specify the absolute path to where the file is located.

The file should contain variable information in the following format:

```
set_parm variable1 value
set_parm variable2 value
set_parm variable3 value
```

If you do not specify this parameter, variable values specified with `setSIMode -insCeltICPreTcl` are used. Specifying values with this parameter overrides variable values specified with `setSIMode`.

`-irDrop {`*`ir_drop_file`*`}`

Uses the supply voltage information from the specified IR drop file to perform noise-on-delay analysis. You must specify the appropriate IR drop file, based on whether you are performing setup or hold analysis.

Note: If you do not specify this parameter, the IR drop file specified with `setSIMode -irDropFile` is used. Specifying an IR drop file with this parameter overrides the file specified with `setSIMode`, and changes the `setSIMode` setting.

`-native` Runs native Signal Integrity analysis.

`-process {180nm | 130nm | 90nm | 65nm | 45nm}`

	Specifies the technology node. This sets up options or arguments that are appropriate for the specified technology node. *Default:* `130`
`-saveCdb`	Creates a cdB model from the SI analysis engine.
`-saveEcho`	Creates an ECHO model from the SI analysis engine.
`-saveXilm`	Creates an XILM model from the SI analysis engine.
`-spef` *`filename`*	Specifies the name of a pre-existing SPEF file to use. The software generates this file if you do not specify one.
`-twf` *`filename`*	Specifies the name of a pre-existing timing window file to use. The software generates this file if you do not specify one. The software uses the timing window information from each input pin of a net to generate the timing window file.
`-update | -noUpdate`	Specifies whether to update the software's timing records after running the `runCeltIC` command. *Default:* `-update`
`-view`	Calls native SI analysis for each analysis view. This parameter supports the concurrent multi-mode multi-corner mode.

setSIMode

```
setSIMode
     [-help]
     [-reset]
     [-acceptableWNS {same | value}]
     [-analysisType { default | pessimistic }]
     [-analyzeNoiseThreshold value]
     [-deltaDelayThreshold value]
     [-detailedReports { true | false }
     [-effortLevel {default | high}]
     [-extractionEngine {detail | incrementalQRC | standaloneQRC}]
     [-fixDelay {true | false}]
     [-fixDRC {true | false}]
     [-fixGlitch {true | false}]
     [-fixHoldIncludeXtalkSetup {true | false}]
     [-insCeltICPostTcl {commands}]
     [-insCeltICPreTcl {variables}]
     [-irDropFile {ir_drop_file_name}]
     [-maxNrIter value]
     [-noiseProcess {130nm | 180nm | 90nm | 65nm | 45nm}]
     [-noiseTwfMode {“ ” | -useCTE | -infSW}]
     [-numQXcpu value]
     [-outputPath dir_path]
     [-runStandAloneNanoRoute {true | false}]
     [-runTrimMetalFill {true | false}]
     [-saveSIFixDB {true | false}]
     [-targetPathGroups {path_group_names}]
     [-usePrevCeltICRunDir directory]
     [-usePrevSpefFile {filename}]
     [-usePrevTwfFile {filename}]
     [-usePrevWdbName {filename}]
     [-wrouteExtraConfig {fileName}]
```

Sets signal integrity configuration parameters used for crosstalk analysis and repair. Parameters set with the `setSIMode` command are then used automatically whenever you run the `optDesign -postRoute -si` or runCeltIC commands.

Use the getSIMode command to display the current settings for the `setSIMode` command.

Parameters

`-acceptableWNS {same | value}`

Specifies the worst negative slack (WNS) that is acceptable for the design.

Default: `same` (that is, the input slack without coupling is equal to the output slack with coupling, or `0`)

`-analysisType { default | pessimistic }`

Specifies the SI analysis mode types. Based on the value specified, the SI analysis engine uses the default or pessimistic settings. Use one of the following values with this parameter:

`default` Resets all the `setSIMode` command parameters to their default value.

`pessimistic` Performs pessimistic SI analysis to generate better correlation results with external signoff analysis tools. When you specify this value, the following settings are applied automatically:

```
setSIMode -insCeltICPreTcl { \
   set_thresh -delta_absolute 1e-12; \
   set_virtual_attacker -mode current
-gtol 0; \
   set_align_mode -mode peak }
```

Important Considerations:

- The pessimistic value should be used before the `optDesign -si` or `timeDesign -si` commands.
- When using the `pessimistic` value, ensure that the design is in multi-mode multi-corner (MMMC) mode and the `setAnalysisMode -analysisType onChipVariation` setting is specified.
- The `pessimistic` value should not be used in combination with the following `setSIMode` parameters:

  `[-analyzeNoiseThreshold` *`value`*`]`

  `[-deltaDelayThreshold` *`value`*`]`

`[-insCeltICPreTcl {`*`variables`*`}]`

`-analyzeNoiseThreshold` *`value`*

Analyzes the noise-on-delay effects when a glitch exceeds the specified threshold percentage.

Default: `20`

Note: When specified, this parameter overrides the threshold value for glitch specified using the `set_thresh` (CeltIC NDC command) within the `-insCeltICPreTcl` parameter.

`-deltaDelayThreshold` *`value`*

Specifies the delta delay threshold for noise-on-delay analysis. Units are in nanoseconds.

Default: `-1`

Note: When specified, this parameter overrides the delay threshold for noise-on-delay analysis specified using the `set_thresh` (CeltIC NDC command) within the `-insCeltICPreTcl` parameter.

`-detailedReports { true | false }`

Controls the details provided through the report files generated during SI analysis. To enable detailed reporting, which is useful for debugging, set this parameter to `true`.

Default: `false`

`-effortLevel {default | high}`

Specifies the level of effort the tool puts forth for crosstalk analysis and repair.

Default: `default`

Important

Option `-effortLevel` is obsolete. The software repairs only SI-induced delays and glitch violations on selected ports by default.

`default` Repairs only SI-induced delays and glitch violations on selected nets.

`high`	**Note:** The `high` parameter value is obsolete. If you specify this value, the software automatically uses the `default` value instead.

`-extractionEngine {detail | incrementalQRC | standaloneQRC}]`

Specifies the RC extraction engine and SI analysis mode.

Important

Option `-extractionEngine` is obsolete and has been replaced by the `-engine` option of the `setExtractRCMode` command.

Default: `detail`

`detail` Uses native RC extraction (detailed) and native SI analysis for crosstalk analysis and repair. The extracted parasitics are used for both static timing analysis (including cross-coupling) and signal integrity analysis to generate more accurate results for a particular process technology.

`incrementalQRC`

Uses native QRC incremental extraction and native SI incremental crosstalk analysis for post-route optimization and SI closure. Extracts the parasitics on only the nets (and neighboring nets) connected to the cells that have been moved, changed, or resized, and nets re-routed due to SI routing changes.

`standaloneQRC`

Uses QRC standalone sign-off extraction and native SI analysis for sign-off crosstalk analysis and repair. This performs a complete sign-off quality extraction and crosstalk analysis and repair of the entire design.

`-fixDelay {true | false}`

Determines whether to fix noise-on-delay during SI optimization. To disable noise-on-delay fixing, set this parameter to `false`.

Default: `true`

`-fixDRC {true | false}`

Enables the flow for performing maximum transition violation fixing. After setting this parameter to `true`, run the `optDesign -postRoute -si` command to perform maximum transition violation fixing using transition information obtained from the transition file (`celtic.slew`) generated during noise analysis.

Note: The transition values will be used for maximum transition violation fixing only and will not affect the timing calculation in Encounter's timing engine. This is because the transition effect is considered by the internal SI engine during delay pushout calculation.

`-fixGlitch {true | false}`

Determines whether to fix glitch violations during SI optimization. To disable fixing of glitch violations, set this parameter to `false`.

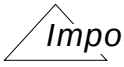
Default: `true`

`-fixHoldIncludeXtalkSetup {true | false}`

Determines whether to account for crosstalk setup timing information while performing SI hold fixing. To account for crosstalk setup timing information while performing SI hold fixing, set this parameter to `true`.

By default (`false`), during SI hold fixing, the software accounts for base timing only.

Default: `false`



When you enable this feature, you will see an increase in run time because the internal SI engine will be called to perform setup noise analysis.

`-help`

Outputs a brief description that includes type and default information for each `setSIMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setSIMode`.

`-insCeltICPostTcl {`*`commands`*`}`

Executes the specified CeltIC NDC commands after the SI analysis engine performs noise analysis. Use the following format to specify the commands:

`{`*`command1 ; command2 ; command3`*`}`

Important

The settings specified with this parameter override the settings that you would have previously set using this parameter.

`-insCeltICPreTcl {`*`variables`*`}`

Changes the default environment variable values to the specified values. Use the following format to specify the variable values:

```
{set_parm variable1 value ; \
 set_parm variable2 value ; \
 set_parm variable3 value ;}
```

Alternatively, you can specify a Tcl file that contains a list of variables. For example,

```
-insCeltICPreTcl {source directory_path/
file_name.tcl}
```

Note: If you specify a file, you must specify the absolute path to where the file is located.

The file should contain variable information in the following format:

```
set_parm variable1 value
set_parm variable2 value
set_parm variable3 value
```

The settings specified with this parameter override the settings that you would have previously set using this parameter.

`-irDropFile {`*`ir_drop_file_name`*`}`

Uses the supply voltage information from the specified IR drop file to perform noise-on-delay analysis. You must specify the appropriate IR drop file, based on whether you are performing setup or hold analysis.

Note: You can only specify this parameter if you are running the `optDesign -postRoute -si` command in sign-off mode (`-extractionEngine standaloneQRC`).

`-maxNrIter` *`value`*

Specifies the maximum number of iterations the tool will perform.

Default: 2

`-noiseProcess {180nm | 130nm | 90nm | 65nm | 45nm}`

Specifies the process name in SI analysis run mode.

Default: `90nm`

Option `-noiseProcess` is obsolete and has been replaced by the `-process` option of the `setDesignMode` command.

`-noiseTwfMode {" " | -infSW | -useCTE}`

Specifies the mode for loading timing window files.

Default: `" "` (empty string)

`" "`	Uses timing window files generated by the Encounter software, based on current delay calculation and static-timing analysis (STA) engine choices.
`-infSW`	States that all aggressors can switch at the same time. The resulting analysis is more pessimistic. Use infinite switching windows when the timing constraints information is not well defined.
`-useCTE`	Uses the timing window files generated by common timing engine (CTE) during SI analysis.

`-numQXcpu`

Specifies the number of processors that standalone QX uses for parasitic extraction. You can use this parameter to distribute the workload to multiple processors to improve on run time.

Default: `1`

Option `-numQXcpu` is obsolete and has been replaced by the `setMultiCpuUsage` and `setDistributeHost` commands.

`-outputPath` *`dir_path`*

Specifies the path of the output directory.

Default: `./result`

Note: When performing timing optimization with the `optDesign` command, the default `./timingReports` directory takes precedence over *`dir_path`* if the `optDesign -si` command parameter was used.

Important

Option `-outputPath` is obsolete. This obsolete option still works in this release, but to ensure compatibility with future releases, remove this option from your script.

`-reset`

Resets parameters to their default values. The `-reset` parameter must be the first parameter specified. If you specify `-reset` by itself, the software resets all `setSIMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-runStandAloneNanoRoute {true | false}`

Runs standalone NanoRoute for analysis and repair.

Default: `false`

Note: You can only specify this parameter if you are running the `optDesign -postRoute -si` command in sign-off mode (`-extractionEngine standaloneQRC`).

`-runTrimMetalFill {true | false}`

Runs the `trimMetalFill` command after routing and during SI fixing to fix timing with crosstalk and crosstalk noise after metal fill.

Default: `false`

`-saveSIFixDB {true | false}`

Saves the design database at the end of each SI fixing flow iteration.

At the end of each SI fixing flow iteration, the Encounter software calls the `saveDesign` command after performing NanoRoute ECO to save the design database. Use this parameter during SI fixing to disable or save intermediate design databases for each iteration of the SI fixing flow. Disabling this parameter (`false`) improves the SI fixing flow run time and saves disk space. To save the design database for each SI fixing flow iteration, set this parameter to `true`.

Default: `false`

`-targetPathGroups {`*`path_group_names`*`}`

Defines the path groups to monitor and optimize during crosstalk repair.

Default: Automatically defines the `reg2reg`, `in2reg`, `reg2out`, and `in2out` path groups

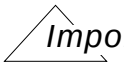
`reg2reg`	Optimizes all paths that begin at a register and end at the next register on the path.
`in2reg`	Optimizes all paths that begin at an input pin and end at the next register on the path.
`in2out`	Optimizes all paths that begin at an input pin and end at an output pin.
`reg2out`	Optimizes all paths that begin at a register and end at an output pin.

`-usePrevCeltICRunDir` *`directory`*

Uses the previous `celtic_ndc` run directory to perform analysis and repair.

`-usePrevSpefFile {`*`filename`*`}`

Uses the previous SPEF file to perform analysis and repair.

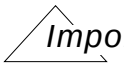


Important

Option `-usePrevSpefFile` is obsolete. This obsolete option still works in this release, but to ensure compatibility with future releases, remove this option from your script.

`-usePrevTwfFile {`*`filename`*`}`

Uses the previous TWF file in sign-off mode only. This parameter supports an external TWF file in the Signal Integrity (SI) flow.

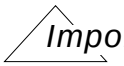


Important

Option `-usePrevTwfFile` is obsolete. This obsolete option still works in this release, but to ensure compatibility with future releases, remove this option from your script.

`-usePrevWdbName {`*`filename`*`}`

Uses the specified previous WRoute database name to perform post-repair routing using WRoute.



Important

Option `-usePrevWdbName` is obsolete. This obsolete option still works in this release, but to ensure compatibility with future releases, remove this option from your script.

`-wrouteExtraConfig {`*`fileName`*`}`

Reads the specified WRoute configuration file to perform post-repair routing using Wroute.

Note: You can only specify this parameter if you are running crosstalk analysis in sign-off mode (`-extractionEngine standaloneQRC`).

Default: `siFix`

Important

Option `-wrouteExtraConfig` is obsolete. This obsolete option still works in this release, but to ensure compatibility with future releases, remove this option from your script.

Example

- The following command resets the `-noiseProcess` parameter to its default value:

  ```
  setSIMode -reset -noiseProcess
  ```

- The following command resets all `setSIMode` parameters to their default values:

  ```
  setSIMode -reset
  ```

Related Topics

- mode.tcl in the *Encounter Flat Implementation Flow Guide*
- SI Closure in the *Encounter Flat Implementation Flow Guide*

viewCeltIC

```
viewCeltIC
     filename
     [-honorPostRouteSiFixFlag]
     [-th threshold]
     [-eco file]
     [-txt file]
```

Opens a GUI in which to view the noise report, and provides hyperlinks to the routing context.

Note: In order to view the noise nets after incremental crosstalk analysis and repair (`-extractionEngine incrementalQRC`), you must first rerun the `runCeltic` command to update the `celtic.eco` and `celtic.txt` files.

Parameters

`-eco` *`file`*	Displays glitch violations recorded in the specified ECO file. *Default:* `./celtic/celtic.eco`
`filename`	Specifies the name of the noise report file. *Default:* `./celtic/celtic.txt`.
`-honorPostRouteSiFixFlag`	Honors nets marked with the signal integrity post-route repair flag.
`-th` *`threshold`*	Specifies the noise threshold percent (`0` to `100`) of VDD. *Default:* `0`
`-txt` *`file`*	Specifies the victim/aggressor net file to be used in the browser. *Default:* `./celtic/celtic.txt`

Example

- The following command opens a GUI in which to view the specified noise report with the noise threshold set at `20`% of VDD:

  ```
  viewCeltIC celtic.txt -th 20
  ```

- The following command opens a GUI in which to view the noise violating nets from the database in memory:

  ```
  viewCeltIC -honorPostRouteSIFixFlag -txt celtic/celtic.txt
  ```

32

Timing Analysis Commands

Path Exception Priorities

The following are the path exception priorities if a path in the design matches more than one path exception:

1. `set_false_path`
2. `set_min_delay`
3. `set_max_delay`
4. `set_multicycle_path`

If there is more than one exception of a given type, for example the `set_multicycle_path` command, the path exception that is more specific has higher priority. A path exception is more specific if it specifies a longer path than the other. For example, the `-from -to` options would have priority over the `-from` option.

If the path has the same number of reference points:

- `-from` option has priority over the `-to` option
- `-to` option has priority over the `-through` option
- `-clock_from` option has priority over the `-clock_to` option

Note: To check for ignored path exceptions, use the `report_path_exceptions -ignored` command.

The following table shows the priorities for path exceptions applied to the same path.

Table 32-1 Path Exception Priorities

Priority	Path Exception
1. (Highest)	`set_false_path`
2.	`set_max_delay -from` *`pin_list`*
3.	`set_max_delay -to` *`pin_list`*
4.	`set_max_delay -through` *`pin_list`*
5.	`set_max_delay -clock_from` *`clkwave_name`*
6.	`set_max_delay -clock_to` *`clkwave_name`*
7.	`set_max_delay` (The most constraining adjustment has the higher priority over less constraining adjustments.)

Table 32-1 Path Exception Priorities

Priority	Path Exception
8.	`set_multicycle_path -from` *`pin_list`*
9.	`set_multicycle_path -to` *`pin_list`*
10.	`set_multicycle_path -through` *`pin_list`*
11.	`set_multicycle_path -clock_from` *`clkwave_name`*
12.	`set_multicycle_path -clock_to` *`clkwave_name`*
13. (Lowest)	`set_multicycle_path` (The most constraining adjustment has the higher priority over less constraining adjustments.)

Examples of Path Exception Priorities

The following are examples of the path exception priorities for a path starting at A, going through B, and ending with C. The pairs of examples are ordered such that the highest priority constraint in one pair has precedence over the highest priority constraint in the next pair.

- In the following pair of constraints, the false path overrides the cycle addition for the path from A through B to C. For all other paths from A, the cycle addition applies.

```
set_false_path -from A -through B -to C
set_multicycle_path -from A 2
```

- In the following pair, the first constraint is applied (two cycles added) because the `-from` option has priority over the `-to` option:

```
set_multicycle_path -from A 3
set_multicycle_path -to C 2
```

- In the following pair, the first constraint is applied (two cycles added) because it specifies a `-through` pin as well as a `-to` pin:

```
set_multicycle_path -through A -to B 3
set_multicycle_path -to B 2
```

- In the following pair, the first constraint is applied (two cycles added) because the `-clock_from` option has precedence over the `-clock_to` option:

```
set_multicycle_path -clock_from CLKA 3
set_multicycle_path -clock_to CLKB 2
```

- In the following pair, the second constraint is applied because the second constraint overwrites the first one:

```
set_multicycle_path -from A 2
```

```
set_multicycle_path -from A 3
```

Bidirectional Pin Defaults

Bidirectional pins have both an input pin and an output pin type. This section explains the default pin type that the system uses for bidirectional pins.

Defaults for Pin (or Port) Constraints

The following defines the pin and port types that are used in the next table.

Table 32-2 Bidirectional Pin (or Port) Type Definitions

Pin Type	Type 1	Type 2
Blackbox pins	out	in
Instance pins	out	in
Netlist (primary) ports	in	out
Hierarchical ports	in	out

Table 32-3 Bidirectional Pin Type Defaults

Pin or Port Constraint	Type	Hierarchical Port
set_input_delay	1	ignored
set_output_delay	2	ignored
set_clock_latency	1	ignored
create_generated_clock	1	used
set_input_transition	1	ignored
set_driving_cell	1	ignored
set_drive	1	ignored

Defaults for Port Constraints

The following table defines the port types that are used in the next table.

Table 32-4 Bidirectional Port Type Definitions

Pin Type	Type 1	Type 2
Netlist (primary) ports	in	out
Hierarchical ports	in	out

Table 32-5 Bidirectional Port Type Defaults

Port Constraint	Type	Hierarchical Port
`set_fanout_load`	2	ignored
`set_load`	2	ignored
`set_max_transition`	2	ignored

Command Descriptions

all_analysis_views

```
all_analysis_views
```

Returns a Tcl list of all defined analysis views in the design.

Use this command after creating multiple analysis views (`create_analysis_view`).

Parameters

None

Examples

- The following commands create analysis views `missionSlow` and `missionFast`:

```
create_analysis_view
    -name missionSlow
    -constraint_mode missionSetup
    -delay_corner dcWCCOM
create_analysis_view
    -name missionFast
    -constraint_mode missionHold
    -delay_corner dcBCCOM
```

- The following command returns a list of the defined analysis views in the design:

```
all_analysis_views
```

  The software returns the following results:

```
{missionSlow missionFast}
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Creating Analysis Views"

all_constraint_modes

```
all_constraint_modes [-active | -active_setup | -active_hold]
```

Returns a Tcl list of all defined constraint modes in the design.

Use this command after creating multiple constraint modes (`create constraint mode`).

Parameters

`-active`	Returns a list of the defined constraint modes associated with all active analysis views.
`-active_setup`	Returns a list of the defined constraint modes associated with active setup analysis views only.
`-active_hold`	Returns a list of the defined constraint modes associated with active hold analysis views only.

Example

- The following commands create constraint modes `missionSetup` and `missionHold`:

```
create_constraint_mode -name missionSetup
    -sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc]
create_constraint_mode -name missionHold
    -sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc
        dont_use.sdc]
```

- The following command returns a list all the defined constraints modes in the design:

```
all_constraint_modes
```

  The software returns the following results:

```
{missionSetup missionHold}
```

- The following command returns a list of all the defined constraint modes associated with the active setup analysis views in the design:

```
all_constraint_modes -active_setup
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Creating Constraint Mode Objects"

- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*

all_delay_corners

```
all_delay_corners
```

Returns a Tcl list of all defined delay calculation corner objects in the design.

Use this command after creating multiple delay calculation corner objects (`create_delay_corner`).

Parameters

None

Examples

- The following commands create the delay calculation corners `dcWCCOM` and `dcBCCOM`:

```
create_delay_corner -name dcWCCOM
    -library_set IsCOM-1V
    -opcond_library stdcell_1V
    -opcond WCCOM
    -rc_corner rc-cworst
create_delay_corner -name dcBCCOM
    -library_set IsCOM-1V
    -opcond_library stdcell_1V
    -opcond BCCOM
    -rc_corner rc-cbest
```

- The following command returns a list of all the delay calculation corners in the design:

```
all_delay_corners
```

  The software returns the following information:

```
{dcWCCOM dcBCCOM}
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Creating Delay Calculation Corner Objects"

all_hold_analysis_views

```
all_hold_analysis_views
```

Returns a Tcl list of all active hold analysis views in the design.

Use this command after setting active hold analysis views for the design (`set_analysis_view`).

Parameters

None

Example

- The following command defines the active setup and hold analysis views for the design:

  ```
  set_analysis_view -setup {missionWCCOM mission2WCCOM}
      -hold {missionBCCOM testBCCOM}
  ```

- The following command returns a list of the active hold analysis views in the design:

  ```
  all_hold_analysis_views
  ```

  The software returns the following results:

  ```
  {missionBCCOM testBCCOM}
  ```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Setting Active Analysis Views"
  - "Guidelines For Setting Active Analysis Views"
  - "Changing the Default Active Analysis View"

all_library_sets

```
all_library_sets
```

Returns a Tcl list of all currently defined library sets in the design.

Use this command after creating multiple library sets (`create_library_set`).

Parameters

None

Examples

- The following commands create the library sets `IsCOM-1V` and `IsCOM-2V`:

  ```
  create_library_set -name IsCOM-1V -timing [list stdcell_F_1V.lib ram_F.lib pad.lib]
  create_library_set -name IsCOM-2V -timing [list stdcell_F_2V.lib ram_F.lib pad.lib]
  ```

- The following command returns a list of all defined library sets in the design:

  ```
  all_library_sets
  ```

  The software returns the following results:

  ```
  {IsCOM-1V IsCOM-2V}
  ```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Creating Library Sets"

all_op_conds

`all_op_conds`

Returns a Tcl list of all operating conditions defined for the design.

Use this command after creating multiple operating conditions (`create_op_cond`).

Parameters

None

Examples

- The following commands create operating conditions called `myPVT1` and `myPVT2` for libraries `IsCOM-1V.lib` and `IsCOM-2V.lib`:

```
create_op_cond -name PVT1
   -library_file IsCOM-1V.lib
   -P 1.0
   -V 1.2
   -T 120


create_op_cond -name PVT2
   -library_file IsCOM-2V.lib
   -P 1.0
   -V 1.2
   -T 20
```

- The following command creates a collection called `myOpConds` that contains the operating conditions `myPVT1` and `myPVT2`:

```
set myOpConds [all_op_conds]
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Creating Virtual Operating Conditions"

all_rc_corners

```
all_rc_corners
```

Returns a Tcl list of all currently defined RC corner objects in the design.

Use this command after creating multiple RC corner objects (`create_rc_corner`).

Parameters

None

Examples

- The following commands create the RC corners `rc-cbest`, `rc-cworst`, and `rc-typical`:

  ```
  create_rc_corner -name rc-cbest -cap_table myTech_bc.CapTbl -T 50
  create_rc_corner -name rc-cworst -cap_table myTech_wc.CapTbl
  create_rc_corner -name rc-typical -cap_table myTech_typ.CapTbl
  ```

- The following command returns a list of all defined RC corners in the design:

  ```
  all_rc_corners
  ```

  The software returns the following results:

  ```
  rc-cbest rc-cworst rc-typical
  ```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Creating RC Corner Objects"

all_setup_analysis_views

```
all_setup_analysis_views
```

Returns a Tcl list of all active setup analysis views in the design.

Use this command after setting active setup analysis views for the design (`set_analysis_view`).

Parameters

None

Example

- The following command defines the active setup and hold analysis views for the design:

  ```
  set_analysis_view -setup {missionWCCOM mission2WCCOM}
      -hold {missionBCCOM testBCCOM}
  ```

- The following command returns a list of the active setup analysis views in the design:

  ```
  all_setup_analysis_views
  ```

  The software returns the following results:

  ```
  {missionWCCOM mission2WCCOM}
  ```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Setting Active Analysis Views"
  - "Guidelines For Setting Active Analysis Views"
  - "Changing the Default Active Analysis View"

buildTimingGraph

```
buildTimingGraph
     [-ignoreNetLoad]
     [-elmore hiFanoutLimit]
```

Builds the static timing model of the design. After running this command, you can create a slack report to output path delays.

Note: Manually building or freeing the timing graph is not required in Encounter. The software builds the timing graph automatically.

Parameters

`–elmore` *`hiFanoutLimit`*

Specifies that high fanout nets use the Elmore model when calculating delays. Any net that has terminals equal to or exceeding this limit will use the Elmore model.

When the pin number of the nets is larger than the *`hiFanoutLimit`*, then the interconnect delay of the nets will be calculated by the elmore delay. It will run faster, but with less accuracy.

`–ignoreNetLoad`

Builds the static timing model without interconnect delays. This parameter gives the raw speed of the design.

Command Order

Use this command after reading the timing library and timing constraints files, and after extracting RC.

calNegSlack

`calNegSlack`

Reports the worst slack, total negative slack, and the number of violating paths.

Command Order

Use this command after running the `buildTimingGraph` command.

Example

The `calNegSlack` command reports the total negative slack, worst negative slack, and the number of violating paths. You can also find this report in the log file.

```
calNegSlack
  Total negative slacks(TNS)= xxx
  Worst negative slacks(WNS)= xxx
  Number of violating paths= xxx
```

check_timing

```
check_timing
     [-type type_list]
     [-verbose]
     [-check_only warning_list]
     [-include_warning warning_list]
     [-exclude_warning warning_list]
     [-view view_name]
     [port_or_pin_list]
     [-tcl_list]
     [-old]
     [{> | >> } file_name[.gz]]
```

Performs a variety of consistency and completeness checks on the timing constraints specified for a design. Valid types are: `clocks`, `clock_clipping`, `constant_collision`, `endpoints`, `input`, `multiple_clocks` and `loops`.

The checks include arrival time and external delay (or required time) for each clock in a multiple clock system. In addition, clock connectivity and data connectivity are checked to make sure the clock or data is propagated as expected. Clock gating points are also reported.

Use this command on generic or mapped netlists. The `check_timing` command considers constants applied to a pin while reporting warnings.

Important

Use the `check_timing` command before using any commands for timing analysis. Warnings are displayed when problems occur. Table 32-6 on page 1569 lists the `check_timing` warnings. For best results, rerun the `check_timing` command after resolving each warning. Redirecting the output to a file is the quickest run method.

Parameters

`-check_only warning_list`

Allows you to specify warning types. For example, the following command lists the specified warnings only:

```
check_timing -check_only {uncons_endpoint
clock_crossing}
```

This list can also include warnings that are not listed by default.

`{> | >>}` *`file_name`*

When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it.

When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the `.gz` extension to the file name.

Default: Report is displayed on standard output without being saved.

`-exclude_warning` *`warning_list`*

Excludes the specified warning types when generating the report. Use this parameter to exclude any of the warnings included in the report by default.

By default, the software includes the following warnings in the report: `async_arr`, `clock_expected`, `loop`, `ideal_clock_waveform`, `master_clock_edge_not_reaching`, `missing_arr`, `missing_drive`, `no_arr`, `no_drive`, `no_gen_clock_source`, `specific_async_arr`, `uncons_endpoint`.

The complete list of warnings and their descriptions is given in Table 32-6 on page 1569.

`-include _warning` *`warning_list`*

Includes the specified warning types when generating the report. Use this parameter to add any of the warnings not included in the report by default.

By default, the software reports all warnings except the following: `clock_clipping_gate`, `const_collision`, `clock_crossing`,`clock_not_propagated`, `missing_drive`, `and multiple_clocks_gate`.

The complete list of warnings and their descriptions is given in Table 32-6 on page 1569.

`-old`

Note: The `-old` parameter is obsolete and is no longer supported.

Prints the timing check warning messages in text format.

`port_or_pin_list`

Lists pins or ports. For example:

```
check_timing -verbose [get_ports in*]
```

`-tcl_list`

Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation.

The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

`-type` *`type_list`*

Limits the checks to the specified type.

Default: All check types are reported.

`clock_clipping`

Warns when there is a problem with clock gating that can result in clock clipping.

Reports the following warnings:

- `clock_clipping_freq`
- `clock_clipping_gate`

`clocks`

Reports the following clock warnings:

- `clock_crossing`
- `clock_expected`
- `clock_not_propagated`
- `ideal_clock_waveform`
- `master_clk_edge_not_reaching`
- `no_gen_clock_source`

`constant_collision`

Warns when there is a constant collision on a net connected to the pin or if there is a contradiction on a pin.

Constant collision occurs when a net is driven simultaneously by conflicting values and a constant contradiction occurs when a constant constraint on a pin conflicts with the driven value.

`endpoints` For output ports, warns if no output delay constraint is applied to the port or if a specific (min/max rise/fall) output delay is missing. It also warns if an output delay constraint is applied with no clock information.

For all endpoints (output ports and register data pins), it warns if any incoming signal is unconstrained.

Also warns if a clock arrives where clock is not expected, or if multiple signals arrive at an endpoint.

`inputs` Reports the following input delay, arrival time assertion, and drive assertion warnings:

- `async_arr`
- `missing_arr`
- `missing_drive`
- `no_arr`
- `no_drive`
- `specific_async_arr`

`loops` Warns when a combinational loop is detected. Reports the loop and the arc used to break the loop. The `-pin` option is ignored in this case.

Also reports loops caused by generated clocks.

`multiple_clocks`

	Reports the following warnings for multiple clocks: ■ `multiple_clocks_gate`
`-verbose`	Gives detailed information about the warnings. When specified, warnings are grouped and sorted by warning type. *Default:* Gives a summary of how many warnings of each type exists.
`-view`	Reports warning(s) that belong to the specified view.

Command Order

Use this command after setting all the constraints but before optimizing your design or generating any timing reports.

Warning Messages

The following table shows the `check_timing` warning messages.

Table 32-6 check_timing Warning Messages

Warning	Description	Comments
`async_arr`	Input delay or arrival time constraint with no clock information (asynchronous).	Input delay or arrival time constraint applied without the `-clock` option. Do not mix an asynchronous clock (@) with a regular clock.
`clock_clipping_gate`	Clock clipping possible due to wrong gate type or wrong trigger for data input.	
`clock_clipping_freq`	Clock clipping possible due to incompatible clock signal and data signal frequencies.	

Table 32-6 check_timing Warning Messages , *continued*

Warning	Description	Comments
`clock_crossing`	Clock domains interact.	Checks clock interactions between multiple clock domains in the design. When you specify this parameter, the Timing Check Crossing Clocks report shows the interaction of clocks in the design and indicates whether the paths between the clocks are `False` or `Partial False.` If a clock launches one or more paths, which are captured by other clocks, the report lists all pairs of crossing clocks. ■ This includes inferred clock gating with different reference and signal clock. ■ If a path has been disabled using the `set_disable_timing` command, crossing clocks on that path are not reported. If all the paths between two clocks are false paths or they are exclusive/asynchronous clocks, the path is marked as `False`. The false path can be defined on clocks, for example, `set_false_path -clock_to CLK` or on pins or ports, such as, `-through buf1/Y`. If only part of paths are set as false paths or exclusive/ asynchronous clocks, the path is marked as `Partial False`.

Table 32-6 check_timing Warning Messages , *continued*

Warning	Description	Comments
		■ If a false path is defined only for one check type (say, setup), the crossing clocks are marked as `Partial False` if the simultaneous setup hold mode is set to `on`. The crossing clocks are marked `False` if the analysis type is setup and the simultaneous setup hold mode is set to `off`. ■ If only one edge (rise/fall) of a particular clock pair has false paths (say, using `-clock_fall`), then the path is marked as a `Partial False`.
`clock_expected`	Clock not found where clock is expected.	Indicates that no clock signal is defined at the clock pin. See `create_clock`.
`clock_not_propagated`	Clock not propagated	Reports ideal clocks reaching the reference pin of a timing check.
`const_collision`	Constant collision	See `set_case_analysis`.
`ideal_clock_waveform`	Clock waveform is ideal.	Reports clocks that are ideal and are not propagated. This warning type is reported by default (unless `-exclude_warning` or `-check_only` parameter is specified). You can use the `set_propagated_clock` command to avoid this warning.

Table 32-6 check_timing Warning Messages , *continued*

Warning	Description	Comments
`loop`	Timing loop found in the design	Indicates that there are timing loops in the design. It also includes loops caused by generated clocks.
`master_clk_edge_not_reaching`	Master clock edge does not reach the generated clock target.	Generated at the target pin of a generated clock when the master clock is broken on the way to the generated clock.
`missing_arr`	Missing specific (rise/fall) input delay or arrival time constraint	Either a rise or fall input delay or arrival time constraint is applied and the other one is missing.
`missing_drive`	Missing specific (rise/fall) drive constraint	Either a rise or fall drive constraint is applied and the other one is missing.
`multiple_clocks_gate`	Multiple clocks arrive at gate output	
`no_arr`	No input delay or arrival time assertion.	See `set_input_delay`.
`no_drive`	No drive constraint	See `set_driving_cell` and `set_drive`.
`no_gen_clock_source`	No clock source found for generated clock	Indicates that the source pin of the clock generated by the `create_clock` command is not driven by a clock source.
`specific_async_arr`	Specific (rise/fall) input delay or arrival time constraint with no clock information (asynchronous)	Either a rise or fall input delay or arrival time constraint is applied without the `-clock` option. Do not mix an asynchronous clock (@) with a regular clock.

Table 32-6 check_timing Warning Messages , *continued*

Warning	Description	Comments
uncons_endpoint	No constrained signal reaching pin.	For output ports: Primary output port has no implicit timing check. Use set_output_delay, set_max_delay, or set_min_delay to create a valid constraint. For register data pins: There is no valid clock reaching the CK pin of the register data pin. Check for missing create_clock for the clock network. There also could be set_false_path exceptions which are invalidating the current data phases at the data pin (or clock phases at the CK pin), preventing the timing check from being evaluated and constraining the timing of the data pin.

Examples

- Consider the following example:

```
in---------|>-----------|>---------out
          buf1         buf2
```

  The Timing Check Crossing Clocks reports `False` or `Partial False` in some of the scenarios given below:

  - `set_input_delay` with respect to `CLK1` on `in` and `set_output_delay` with respect to `CLK2` on `out`:

```
From Clock         Crossing Clocks
-----------------------------------------
CLK1               CLK2
```

  - `set_input_delay` with respect to `CLK1` on `in`, `set_output_delay` with respect to `CLK2` on `out`, and `set_false_path -to CLK2` (clock-based false path):

```
From Clock         Crossing Clocks
```

```
-----------------------------------------
CLK1              CLK2  (False)
```

- If the `set_false_path` becomes `set_false_path -rise_to CLK2`, the following is reported:

```
From Clock        Crossing Clocks
-----------------------------------------
CLK1              CLK2  (Partial False)
```

- The following command displays the default summary report:

```
check_timing
```

```
+-------------------------------------------------------------------------+
|                              TIMING CHECK SUMMARY                       |
|-------------------------------------------------------------------------|
|     Warning        |             Warning Description            | Number |
|                    |                                            |   of   |
|                    |                                            |Warnings|
|-------------------------------------------------------------------------|
|clock_expected      | Clock not found where clock is expected    |      1 |
|no_drive            | No drive assertion                         |      3 |
+-------------------------------------------------------------------------+
```

- The following command displays a specified type of warning using the `-type` option if a timing loop is found in the design:

```
check_timing -type loops
```

```
 +-----------------------------------------------------+
|                TIMING CHECK SUMMARY                 |
|-----------------------------------------------------|
| Warning |       Warning Description       |  Number  |
|         |                                 |    of    |
|         |                                 | Warnings |
|---------+---------------------------------+----------|
| loop    | Timing loop found in the design |        1 |
 +-----------------------------------------------------+
```

- The following command provides more information about problems and prints a summary using the `-verbose` option:

```
check_timing -verbose
```

TIMING CHECK SUMMARY

Warning	Warning Description	Number of Warnings
clock_expected	Clock not found where clock is expected	1
missing ext	Missing specific (rise/fall) external delay or required time assertion	1
multiple_clocks_gate	Multiple clocks at gate output	1
no_drive	No drive assertion	3

TIMING CHECK DETAIL

Pin	Warning
clkA	No drive assertion
clkA	No input delay or arrival time assertion
clkB	No drive assertion
f1/CP	Clock not found where clock is expected
f1/CP	Data signal (@ lead) found where clock is expected
f1/CP	Unconstrained signal (@ lead) arriving at end point
f1/D	Unconstrained signal (ideal lead) arriving at end point
in	No drive assertion
out	No external delay or required time assertion
out1	No external delay or required time assertion
UC02/X	Multiple clocks ({Iclf1} on pin UC02/A {Iclk2} on pin UC02/B no constants set on pin {UC02/S} cell SCJSL12010) at gate output

TIMING CHECK LOOP

Snipped Arcs		Timing Loop Details
From Pin	To Pin	
g1/Z	g1/A	g1/A
		g1/Z

- The following example shows a timing check report when the software is in multi-mode multi-corner mode. The report identifies the warning messages for each active analysis view.

Example 32-1

```
+-------------------------------------------------------------------------------+
|                              TIMING CHECK DETAIL                              |
|-------------------------------------------------------------------------------|
|     Pin          |             Warning                  |        View       |
|------------------+--------------------------------------+-------------------|
|  BG_scan_in      | No drive assertion                   | core+worst-rcTyp  |
|  BG_scan_in      | No drive assertion                   | core+best-rcTyp   |
|  BG_scan_in      | No drive assertion                   | io+typ-rcMax      |
|  BG_scan_in      | No input delay or arrival time       | core+worst-rcTyp  |
|                  | assertion                            |                   |
|  BG_scan_in      | No input delay or arrival time       | core+best-rcTyp   |
|                  | assertion                            |                   |
|  BG_scan_out     | No external delay or required time   | core+best-rcTyp   |
|                  | assertion                            |                   |
|  BG_scan_out     | No external delay or required time   | core+worst-rcTyp  |
|                  | assertion                            |                   |
```

checkTimingLibrary

```
checkTimingLibrary
     [-outfile fileName]
     [-cellName cellName]
     [-library timingLibraryName]
     [-checkPower]
     [-reportMissingPowerOnly]
```

Checks the contents of the timing library and reports any inconsistencies in a log file.

You can use the `checkTimingLibrary` command after importing the design and timing libraries.

Parameters

`-cellName` *`cellName`*
: Checks if the timing is defined for all cells with the specified name, in all timing libraries.

  Note: In general, the `-cellName` parameter is optional. However, if you specify the `-library` parameter, then you *must* specify `-cellName`.

`-checkPower`
: Checks if power constructs are defined for the cells, such as internal_power, leakage_power, or ecsm_power.

  Note: In general, the `-checkPower` parameter is optional. However, if you specify `-reportMissingPowerOnly`, then you *must* specify `-checkPower`. If you use the `-library` parameter, you do not have to specify the `-checkPower` parameter, because when a library-specific cell is given as input, the software automatically checks for power constructs.

  Default: Only checks if timing is defined for the cells.

`-library` *`timingLibraryName`*
: Specifies the timing library in which to check a particular cell. You must specify the `-cellName` parameter with `-library`, to specify the particular library cell to check.

  Note: If you use the `-library` parameter, you do not have to specify the `-checkPower` parameter, because when a library-specific cell is given as input, the software automatically checks for power constructs.

`-outfile` *`fileName`*
: Writes the report to the specified file.

`-reportMissingPowerOnly`

Reports only the cells that do not have internal_power (at pin level), or leakage_power defined.

You must specify the `-checkPower` parameter with the `-reportMissingPowerOnly` parameter.

Examples

- The following command checks if timing is defined for all cells that have been read into the design, and writes a report to the log file:

  ```
  checkTimingLibrary
  ```

- The following command checks if timing is defined for all cells that have been read into the design, and writes a report to a file named `CheckTiming1`:

  ```
  checkTimingLibrary -outfile CheckTiming1
  ```

- The following command checks if timing and power constructs are defined for all cells that have been read into the design, and writes the report to the log file:

  ```
  checkTimingLibrary -checkPower
  ```

- The following command checks if timing and power constructs are defined for all cell that have been read into the design, and writes the report to a file named `CheckTiming2`:

  ```
  checkTimingLibrary -outfile CheckTiming2 -checkPower
  ```

- The following command checks if timing is defined for all cells with the name `A01` in all timing libraries, and writes the report to a file named `CheckTiming3`:

  ```
  checkTimingLibrary -outfile CheckTiming3 -cellName A01
  ```

- The following command checks if timing and power constructs are defined for all cells with the name `A01`, in all timing libraries, and writes the report to a file named `AllA01Check`:

  ```
  checkTimingLibrary -outfile AllA01Check -cellName A01 -checkPower
  ```

- The following command checks if timing and power constructs are defined for the cell with the name `A01` in the timing library `minFile1.lib`, and writes the report to a file named `A01Check`:

  ```
  checkTimingLibrary -outfile A01Check -cellName A01 -library minFile1.lib
  ```

  Note: When you use the `-library` parameter, you do not have to specify the `-checkPower` parameter, because when a library-specific cell is given as input, the software automatically checks for power constructs.

- The following command checks if the internal_power and/or leakage_power constructs are defined for all cells that have been read into the design, and reports only those cells

that do not have the constructs defined. The software writes the report to a file named `MissingPower`:

```
checkTimingLibrary -outfile MissingPower -checkPower -reportMissingPowerOnly
```

clearClockDomains

```
clearClockDomains
     [-fromType {input | register | flopOrLatch}]
     [-toType {output | register | flopOrLatch}]
```

Clears clock domain settings. Use this command to incrementally clear the selected clock domain settings.

Parameters

`-fromType {input | register | flopOrLatch}`

Includes paths based on the source point of paths.
Default: All source points are used

The source point can be one of the following:

`input`	Includes paths that originate at the inputs.
`register`	Includes paths that originate at the registers and macros.
`flopOrLatch`	Includes paths that originate at flip-flops and latches.

`-toType {output | register | flopOrLatch}`

Includes paths based on the destination points of paths.
Default: All destination points are used

The destination point can be one of the following:

`output`	Includes paths that terminate at the outputs.
`register`	Includes paths that terminate at the registers and macros.
`flopOrLatch`	Includes paths that terminate at flip-flops and latches.

Command Order

Use this command after setting clock domain information.

Related Topics

- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Partitions in the *Encounter Hierarchical Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Top-Level in the *Encounter Hierarchical Implementation Flow Guide*
- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- SI Closure in the *Encounter Flat Implementation Flow Guide*
- Partition SI Closure in the *Encounter Hierarchical Implementation Flow Guide*
- Top-Level SI Closure in the *Encounter Hierarchical Implementation Flow Guide*

create_analysis_view

```
create_analysis_view
     -name viewName
     -constraint_mode modeName
     -delay_corner dcCornerObj
```

Creates an analysis view object that associates a delay calculation corner with a constraint mode. An analysis view object provides all of the information necessary to control a single multi-mode multi-corner analysis.

After creating analysis views, use the set_analysis_view command to specify which views to use for setup and hold optimization or timing analysis.

Use this command after creating constraint modes (create_constraint_mode) and delay calculation corners (create_delay_corner).

Parameters

`-name` *viewName*
: Specifies the name of the analysis view being defined.

`-constraint_mode` *modeName*
: Specifies the name of the constraint mode to associate with this analysis view. A constraint mode groups a set of constraints to be used with any given analysis. To create the constraint mode, use the create_constraint_mode command.

`-delay_corner` *dcCornerObj*
: Specifies the name of the delay calculation corner object to associate with this analysis view. A delay calculation corner object contains all of the information needed to control delay calculation for a specific analysis view. To create a delay calculation corner, use the create_delay_corner command.

Examples

- The following command creates an analysis view called `missionSlow` using the constraint mode `missionSetup` and the delay calculation corner `dcWCCOM`:

```
create_analysis_view
    -name missionSlow
    -constraint_mode missionSetup
    -delay_corner dcWCCOM
```

- The following command creates an analysis view called `missionFast` using the constraint mode `missionHold` and the delay calculation corner `dcBCCOM`:

```
create_analysis_view
    -name missionFast
    -constraint_mode missionHold
    -delay_corner dcBCCOM
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Configuring the Setup for Multi-Mode Multi-Corner Analysis"
  - "Creating Analysis Views"
  - "Setting Active Analysis Views"
  - "Guidelines For Setting Active Analysis Views"
  - "Changing the Default Active Analysis View"
- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- mmmc.tcl in the *Encounter Hierarchical Implementation Flow Guide*

create_constraint_mode

```
create_constraint_mode
     -name modeName
     -sdc_files {file1.sdc file2.sdc ...}
     [-ilm_sdc_files {file1.sdc file2.sdc ...}]
```

Associates a list of SDC constraint files with a specified constraint mode name, for multi-mode multi-corner analysis. This constraint mode name can be referred to later when creating analysis views. A constraint mode defines one of possibly many different functional, test, or Dynamic Voltage and Frequency Scaling (DVFS) modes of a design. SDC files can be shared by many different constraint modes, and the same constraint mode can be associated with multiple analysis views.

SDC files typically contain timing analysis information, such as the clock specifications, case analysis constraints, I/O timings, and path exceptions that make each mode unique. If the ILM flow is being used, the timing system utilizes the files specified with `-ilm_sdc_files` during the flatten ILM mode (`flattenIlm`); otherwise, the system uses the files specified with `-sdc_files`.

Parameters

`-ilm_sdc_files {file1.sdc file2.sdc ...}`
: Specifies a Tcl list of ILM SDC files to be included in the constraint mode.

`-name modeName`
: Specifies the name of the constraint mode to be created.

`-sdc_files {file1.sdc file2.sdc ...}`
: Specifies a Tcl list of SDC files to be included in the constraint mode.

Examples

- The following command groups the SDC files `io.sdc`, `mission1-clks.sdc`, and `mission1-except.sdc` to create a mode object named `missionSetup`:

  ```
  create_constraint_mode -name missionSetup
      -sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc]
  ```

- The following command groups the SDC files `io.sdc`, `mission1-clks.sdc`, `mission1-except.sdc`, and `dont_use.sdc` to create a mode object named `missionHold`:

  ```
  create_constraint_mode -name missionHold
  ```

```
    -sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc
         dont_use.sdc]
```

- The following command groups the SDC files `test-io.sdc`, `test-clks.sdc`, and `test-except.sdc` to create a mode object named `testHold`:

```
create_constraint_mode -name testHold
    -sdc_files [list test-io.sdc test-clks.sdc test-except.sdc]
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Configuring the Setup for Multi-Mode Multi-Corner Analysis"
  - "Creating Constraint Mode Objects"
- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- mmmc.tcl in the *Encounter Hierarchical Implementation Flow Guide*

create_delay_corner

```
create_delay_corner
     -name delayCornerName

     { -library_set libSetObj
        [-opcond_library libName]
        [-opcond opcondName]
        [-rc_corner rcCornerObj]
        [-irdrop_file list_of_files]

     | -late_library_set libSetObj
       -early_library_set libSetObj
        [-late_opcond_library libName -early_opcond_library libName]
        [-late_opcond opcondName -early_opcond opCondName]
        [-late_irdrop_file list_of_files -early_irdrop_file list_of_files]
     }
```

Creates a named delay calculation corner object that can be referenced later when creating an analysis view. A delay calculation corner provides all of the information necessary to control delay calculation for a specific view. Each corner contains information on the libraries to use, the operating conditions with which the libraries should be accessed, and the RC extraction parameters to use for calculating parasitic data. Delay corner objects can be shared by multiple top-level analysis views.

Use separate delay calculation corners to define Best-Case and Worst-Case differences. Use the `-early_*` and `-late_*` parameters within a single delay calculation corner to control on-chip variation.

Note: A single delay calculation corner object specifies the delay calculation rules for the entire design. If a design includes power domains, the delay calculation corner can contain domain-specific subsections that specify the required operating condition information, and any necessary timing library rebinding for the power domain. Use the `update_delay_corner` command to add a power domain definition to a delay calculation corner.

Parameters

`-early_irdrop_file` *`list_of_files`*

Specifies the list of IR drop files to apply when calculating early arrival times at a single delay corner.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

`-early_library_set` *`libSetObj`*

Specifies the library set to associate with this delay corner object for calculating early arrival times at a single delay corner.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

`-early_opcond` *`opcondName`*

Specifies the operating condition to use for calculating early arrival times at a single delay corner.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

Default: Each library in the early library set uses its own default operating condition.

`-early_opcond_library` *`libName`*

Specifies the internal library name for the library in which the early operating condition is defined. This is *not* the library file name.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

Default: The software searches the early library set for the specified operating condition (`-early_opcond`), starting with the master library.

`-irdrop_file` *`list_of_files`*

Specifies the list of IR drop files to apply to both early and late delay calculation for this delay corner object.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

`-late_irdrop_file` *`list_of_files`*

Specifies the list of IR drop files to apply when calculating late arrival times at a single delay corner.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

`-late_library_set` *`libSetObj`*

Specifies the library set to associate with this delay corner object for calculating late arrival times at a single delay corner.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

`-late_opcond` *`opcondName`*

Specifies the operating condition to use for calculating late arrival times at a single delay corner.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

Default: Each library in the late library set uses its own default operating condition.

`-late_opcond_library` *`libName`*

Specifies the internal library name for the library in which the late operating condition is defined. This is *not* the library file name.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

Default: The software searches the late library set for the specified operating condition (`-late_opcond`), starting with the master library.

`-library_set` *`libSetObj`*

Specifies the library set to associate with this delay corner object.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

`-name` *`delayCornerName`*

Specifies the name for the delay corner object being created.

`-opcond` *`opcondName`* Specifies the operating condition to use for setup and hold analysis.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

Default: Each library in the library set uses its own default operating condition.

`-opcond_library` *`libName`*

Specifies the internal library name for the library in which the operating condition is defined. This is *not* the library file name.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

Default: The software searches the library set for the specified operating condition (`-opcond`), starting with the master library.

`-rc_corner` *`rcCornerObj`*

Specifies the RC corner object to associate with this delay corner object.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

Examples

- The following command creates a delay calculation corner called `dcWCCOM`. This corner uses the libraries from `IsCOM-1V`, sets the operating condition to `WCCOM`, as defined in the `stdcell_1V` timing library, and uses the `rc-cworst` RCcorner:

```
create_delay_corner -name dcWCCOM
    -library_set IsCOM-1V
    -opcond_library stdcell_1V
    -opcond WCCOM
    -rc_corner rc-cworst
```

- The following command creates a delay calculation corner called `dcBCCOM`.This corner uses the libraries from `IsCOM-1V`, sets the operating condition to `BCCOM`, as defined in the `stdcell_1V` timing library, and uses the `rc-cbest` RC corner:

```
create_delay_corner -name dcBCCOM
```

```
    -library_set IsCOM-1V
    -opcond_library stdcell_1V
    -opcond BCCOM
    -rc_corner rc-cbest
```

- The following commands create best-case (`dcBBCOM-OCV`) and worst-case (`dcWCCOM-OCV`) delay calculation corners. The OCV variance in each corner is due to different early and late libraries and operating conditions within each delay corner.

```
create_delay_corner -name dcBCCOM-OCV
    -late_library_set BCCOM-SLOW
    -early_library_set BCCOM-FAST
    -late_opcond_library BCCOM-SLOW
    -early_opcond_library BCCOM-FAST
    -late_opcond SLOW
    -early_opcond FAST
create_delay_corner -name dcWCCOM-OCV
    -late_library_set WCCOM-SLOW
    -early_library_set WCCOM-FAST
    -late_opcond_library WCCOM-SLOW
    -early_opcond_library WCCOM-FAST
    -late_opcond SLOW
    -early_opcond FAST
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Configuring the Setup for Multi-Mode Multi-Corner Analysis"
  - "Creating Delay Calculation Corner Objects"
- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- timing.tcl in the *Encounter Hierarchical Implementation Flow Guide*

create_library_set

```
create_library_set
     -name libSetName
     -timing {lib1.lib lib2.lib lib3.lib ...}
     [-si {lib1.cdb lib2.cdb lib3.udn lib4.udn...}]
```

Associates a TCL list of timing and cdB/UDN libraries with a specified library set name. Library sets allow a group of library files to be treated as a single entity so that higher-level descriptions (delay calculation corners) can simply refer to the library configuration by name. The same library set can be referenced multiple times by different delay calculation corners.

Note: You must have a set of timing libraries specified in the configuration file for the software to initialize properly. When configuring your multi-mode multi-corner environment, you can choose to specify these libraries in your own custom library set, or refer to them through the two library set definitions that are automatically initialized from the configuration file: `default_libs_min` and `default_libs_max`.

Important

The order in which you define timing libraries is important. The software considers the first library you specify in the list as the master library, with each successive library having a lower priority.

Parameters

`-name` *`libSetName`*
Specifies the name for the library set being created. This name is used to associate the library list with a specific delay calculation corner.

`-si {`*`lib1`*`.cdb `*`lib2`*`.cdb `*`lib3`*`.udn `*`lib4`*`.udn…}`
Specifies cdB libraries and/or user-defined noise (UDN) models to include in the library set. The cdB and/or UDN libraries are required for performing signal integrity analysis.

`-timing {`*`lib1`*`.lib `*`lib2`*`.lib `*`lib3`*`.lib …}`
Specifies the timing libraries to include in the library set.

Examples

- The following command creates a library set that associates timing libraries and cdB libraries with a nominal voltage of 1 volt with the library name `IsCOM-1V`:

```
create_library_set
    -name IsCOM-1V -timing [list stdcell_F_1V.lib ram_F.lib pad.lib]
    -si [list stdcell_F_2.cdb ram_F.cdb pad.cdb]
```

- The following command creates a library set that can be used to specify the 2-volt version of a core cell library, which could be associated later with a 2-volt MSMV power domain specified in the design:

```
create_library_set
    -name IsCOM-2V
    -timing [list stdcell_F_2V.lib]
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Configuring the Setup for Multi-Mode Multi-Corner Analysis"
  - "Creating Library Sets"
- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- mmmc.tcl in the *Encounter Hierarchical Implementation Flow Guide*

create_op_cond

```
create_op_cond
     -name virtualOpcondName
     -library_file libraryFileName
     -P processValue
     -V voltageValue
     -T temperatureValue
```

Creates a set of virtual operating conditions in the specified library without actually modifying the library. These virtual operating conditions can then be referenced by a delay calculation corner (`create_delay_corner`) as if they actually exist in the library.

Parameters

`-library_file` *`libraryFileName`*

Specifies the library for which to create the virtual operating conditions.

`-name` *`virtualOpcondName`*

Specifies the name for the operating condition being created.

`-P` *`processValue`* Specifies the process value for the operating condition.

`-T` *`temperatureValue`*

Specifies the temperature value for the operating condition.

`-V` *`voltageValue`* Specifies the voltage value for the operating condition.

Examples

- The following command creates a virtual operating condition called `PVT1` for the library `IsCOM-1V.lib`:

```
create_op_cond -name PVT1
   -library_file IsCOM-1V.lib
   -P 1.0
   -V 1.2
   -T 120
```

- The following command creates a virtual operating condition called `PVT2` for the library `IsCOM-2V.lib`:

```
create_op_cond -name PVT2
   -library_file IsCOM-2V.lib
   -P 1.0
   -V 1.2
   -T 20
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Configuring the Setup for Multi-Mode Multi-Corner Analysis"
  - "Creating Virtual Operating Conditions"

create_rc_corner

```
create_rc_corner
     -name rcCornerName
     [-cap_table capTableFile]
     [-T rcTemperatureValue]
     [-default_res_factor float]
     [-detailed_res_factor float]
     [-res_factor float]
     [-default_cap_factor float]
     [-detailed_cap_factor float]
     [-xcap_factor float]
     [-qx_lib_file fileName]
     [-qx_tech_file fileName]
     [-qx_conf_file fileName]
```

Creates a named RC corner object that can be referenced later when creating a delay calculation corner object. An RC corner object provides the software with all of the information necessary to properly extract, annotate, and use the RCs for delay calculation.

Note: You must use the RC corner scaling attributes when running the software in multi-mode multi-corner analysis mode. Scaling factors set using the `setRCFactor` command are ignored in this mode. However, you can still use the `setRCFactor` command to set scaling factors when running Single or Best-Case Worst-Case (BcWc) timing analysis. Use the `setExtractRCMode` command to set the extraction mode for all RC corner objects in the design.

Parameters

`-cap_table` *`capTableFile`*

Specifies the capacitance table to use for extraction when using this RC corner.

Default: The Encounter extraction system uses internal rules for extracting resistances and capacitances.

`-default_cap_factor` *`float`*

Specifies the capacitance scale factor for RC Extraction in default mode.

Default: `1.0`

`-default_res_factor` *`float`*

Specifies the resistance scale factor for correlation, when the software is using default extraction.

`-detailed_cap_factor` *`float`*

Specifies the capacitance scale factor for RC extraction in detail mode.

Default: `1.0`

`-detailed_res_factor` *`float`*

Specifies the resistance scale factor for correlation, when the software is using detailed extraction.

`-name` *`rcCornerName`*

Specifies the name for the RC corner object being created. You can choose any name for an RC corner object, but it is recommended that you use a name that can be easily recognized as referring to a specific RC corner (for example, rc-min, rc-max, c-min, c-max, and so on).

`-res_factor` *`float`*

Note: The `-res_factor` parameter is obsolete. The parameter still works in this release, but to ensure compatibility with future releases, use the `-default_res_factor` or `-detailed_res_factor` parameter instead.

Specifies the resistance scale factor for correlation.

Default: `1.0`

`-qx_conf_file` *`fileName`*

Specifies the name of a user-created configuration file to use with QRC signoff extraction.

The configuration file contains commands and variables that define the extraction environment (technology filename, library name, and so on), specifies the net(s) to extract and how to extract them, controls the resistance and capacitance extraction, and specifies the extraction outputs.

`-qx_lib_file` *`fileName`*

Specifies the name of the cell library file used by QRC extraction to perform sign-off RC extraction. You can either specify an existing library filename, or use the `runLibGen` command to generate a library file.

`-qx_tech_file` *`fileName`*

Specifies the name of an QRC technology (`.tch`) file used by CCE or QRC sign-off RC extraction.

The technology file is generated by TechGen, a built-in functionality of QRC. For most technologies you can get this file from the technology vendors such as TSMC and UMC. For technologies that are not available through the technology vendors, you enter the fabrication process information into an ASCII-format interconnect technology (ICT) input file. The ICT file is then used as input to IceCaps, which in turn generates the technology file.

For more information on creating the ICT files, see Appendix A "Creating the ICT File", in the *Encounter User Guide*.

`-T` *`rcTemperatureValue`*

Specifies the temperature, in units of Celsius, to use to derate resistance values.

Use this parameter when you want to override the default temperature specified in the capacitance table.

Note: The software does not require that the temperature specified by the PVT operating condition at the delay calculation corner level be the same as the RC nominal or user-specified temperature value.

Default: The Encounter extraction system uses the default temperature of 25 degrees Celsius, unless it is overridden by the capacitance table itself.

`-xcap_factor` *`float`* Specifies the cross-coupling capacitance scale factor.

Default: `1.0`

Examples

- The following command creates an RC corner called `rc-cbest` that uses the capacitance table `myTech_bc.CapTbl`, and derates the resistance values based on the temperature of 50 Celsius:

```
create_rc_corner -name rc-cbest -cap_table myTech.4_bc.CapTbl -T 50
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Configuring the Setup for Multi-Mode Multi-Corner Analysis"
  - "Creating RC Corner Objects"
- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- mmmc.tcl in the *Encounter Hierarchical Implementation Flow Guide*

createUserDisableForCombLoopBreak

```
createUserDisableForCombLoopBreak
     -input_file fileName
     -outfile fileName
```

Generates SDC file containing constraints for combinational loop breaking. The command selects the disable arc (s) with flag set to 1 from the PrimeTime report file, and generates the equivalent SDC commands in an output file.

You can load the output SDC file at any time during an Encounter session. To load the file, either include it in the `.config` file or use the `loadTimingCon -incr` command.

Parameters

`-input_file fileName`

Specifies the disabled time arcs report that you generate using PrimeTime. To generate the report, use the following command in PrimeTime:

```
report_disbale_timing -nosplit > pt_disable_arcs.rpt
```

`-outfile fileName` Specifies the name of the output SDC file.

Command Order

Use this command after generating the PrimeTime disable arcs report.

Example

The following command generates an `encCombLoopBreak.sdc` file when the input is a `pt_disable_arcs.rpt` PrimeTime file:

```
createUserDisableForCombLoopBreak -input_file pt_disable_arcs.rpt
           -outfile encCombLoopBreak.sdc
```

The contents of `pt_disable_arcs.rpt` file are as follows:

```
      Cell or Port          From             To          Sense  Flag Reason
 ------------------------------------------------------------------------------
           :
  uchip_core/u_s5_power   oLpcSd_4_   orig_oSmAcpiSd_4_     *         1
```

```
            :
```

The contents of output file `encCombLoopBreak.sdc` are as follows:

```
:
set_disable_timing -from {oLpcSd_4_} -to {orig_oSmAcpiSd_4_}
                [get_cells {uchip_core/u_s5_power}]:
```

freeTimingGraph

```
freeTimingGraph
```

Resets the timing model values after you use the `buildTimingGraph` command.

Note: Manually building or freeing the timing graph is not required in Encounter. The software builds the timing graph automatically.

Parameters

None

get_analysis_view

```
get_analysis_view
     viewName
     {-constraint_mode | -delay_corner}
```

Returns the constraint mode or delay calculation corner associated with the specified analysis view.

Use this command after creating at least one analysis view (`create_analysis_view`).

Parameters

`-constraint_mode`	Returns the constraint mode associated with the specified analysis view.
`-delay_corner`	Returns the delay calculation corner associated with the specified analysis view.
`viewName`	Specifies the name of the analysis view to query.

Examples

- The following command creates the analysis view `missionSlow`:

  ```
  create_analysis_view
      -name missionSlow
      -constraint_mode missionSetup
      -delay_corner dcWCCOM
  ```

- The following command returns the delay calculation corner associated with the `missionSlow` analysis view:

  ```
  get_analysis_view missionSlow -delay_corner
  ```

  The software returns the following result:

  ```
  dcBCCOM
  ```

get_capacitance_unit

```
get_capacitance_unit
```

Returns the session capacitance unit in picofarads.

Example

The following command returns the session capacitance unit:

```
get_capacitance_unit
1.000000pf
```

Related Information

`setLibraryUnit`

get_constant_for_timing

```
get_constant_for_timing
     [-bidi_input | -bidi_output]
     pin_name
```

Queries the design database for the state of the specified pin or the state propagated through the combinational logic cone to that pin. The returned value is `0` for logic 0, `1` for logic 1, `X` for undefined, and `Z` for tristate.

Note: Constants on hierarchical ports are not computed.

Tip

The `get_constant_for_timing` command returns the state for one pin only. If you want to know the state on all ports that have a constant set, use the `-type constant` option with the `report_ports` command. For example: `report_ports -type constant`.Use the `-tcl_list` option to process the list in Tcl. See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

Parameters

`-bidi_input | -bidi_output`

Returns the constant that is set on the input or output part of bidirectional pins.

Default: Returns the constant on the output part.

`pin_name`

Specifies the name of the single pin to query for pin state. The pin must be an instance pin or primary I/O port.

Examples

- The following command sets the value of the specified pin(s) to either a 1 or 0 for use by the timing engine:

  ```
  set_case_analysis 0 J_block/zbuf0/A
  ```

- The following command queries the design database for the state of the specified pin or the state propagated through the combinational logic cone to that pin:

  ```
  get_constant_for_timing J_block/zbuf0/A
  ```

Related Information

report_ports

set_case_analysis

report_case_analysis

get_constraint_mode

```
get_constraint_mode
     modeName
     {-sdc_files | -ilm_sdc_files}
```

Returns a Tcl list of the regular SDC constraint files, or the ILM flow SDC constraint files associated with the specified constraint mode.

Use this command after creating constraint modes (`create_constraint_mode`).

Parameters

`-ilm_sdc_files`	Returns the ILM flow SDC files associated with the specified constraint mode.
`modeName`	Specifies the name of the constraint mode to query.
`-sdc_files`	Returns SDC files associated with the specified constraint mode.

Example

- The following command creates the constraint mode `missionSetup`:

  ```
  create_constraint_mode -name missionSetup
      -sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc]
  ```

- The following command returns the SDC constraint files associated with `missionSetup`:

  ```
  get_constraint_mode missionSetup -sdc_files
  ```

  The software returns the following Tcl list:

  ```
  {io.sdc mission1-clks.sdc mission1-except.sdc}
  ```

get_delay_corner

```
get_delay_corner
     delayCornerName
     {  -power_domain_list
      | -power_domain powerDomainName -delayCornerAttribute
      | -delayCornerAttribute
     }
```

Returns attribute information for the specified delay calculation corner object, or for one of its power domains.

Use this command after creating at least one delay calculation corner object (`create_delay_corner`). The delay calculation corner must include a power domain definition in order to query power domain information (`update delay corner`).

Parameters

`-delayCornerAttribute`

Returns the setting for the specified delay calculation corner attribute or power domain attribute.

You can query one of the following delay calculation corner attributes: `-library_set`, `-opcond_library`, `-opcond`, `-rc_corner`, `-irdrop_file`, `-late_library_set`, `-early_library_set`, `-late_opcond_library`, `-early_opcond_library`, `-late_opcond`, `-early_opcond`, `-late_irdrop_file`, or `-early_irdrop_file`.

You can query one of the following power domain attributes: `-library_set`, `-opcond_library`, `-opcond`, `-irdrop_file`, `-late_library_set`, `-early_library_set`, `-late_opcond_library`, `-early_opcond_library`, `-late_opcond`, `-early_opcond`, `-late_irdrop_file`, or `-early_irdrop_file`.

For attribute information, see the `create_delay_corner` command.

`delayCornerName` Specifies the name of the delay calculation corner to query.

`-power_domain` *`powerDomainName`*

Specifies the name of the power domain you want to query.

`-power_domain_list` Returns a list of all the power domains associated with the specified delay calculation corner.

Examples

- The following command creates the delay calculation corner `dc1:`

```
create_delay_corner -name dc1
    -library_set libs-2volt
    -opcond_library delayvolt_2V
    -opcond slow_2V
    -rc_corner rc-cworst
```

- The following command returns the operating condition for the delay calculation corner `dc1`:

```
get_delay_corner dc1 -opcond
```

  The software returns the following results:

```
slow_2V
```

- The following commands add definitions for power domains `domain-3V` and `domain-5V` to the delay calculation corner `dc1`:

```
update_delay_corner -name dc1
    -power_domain domain-3V
    -library_set libs-3volt
    -opcond_library delayvolt_3V
    -opcond slow_3V
update_delay_corner -name dc1
    -power_domain domain-5V
    -library_set libs-5volt
    -opcond_library delayvolt_5V
    -opcond slow_5V
```

- The following command returns a list of all the power domains associated with the delay calculation corner `dc1`:

```
get_delay_corner dc1 -power_domain_list
```

  The software returns the following results:

```
{domain-3V domain-5V}
```

- The following command returns the name of the library set associated with the power domain `domain-5V`:

```
get_delay_corner dc1 -power_domain domain-5V -library_set
```

  The software returns the following results:

```
libs-5volt
```

get_interactive_constraint_modes

```
get_interactive_constraint_modes
```

Returns a list of the multi-mode multi-corner constraint modes that are in interactive constraint entry mode.

When the software is in interactive constraint entry mode (`set_interactive_constraint_modes`), any timing constraints you specify take effect immediately on all active analysis views that are associated with these constraint modes.

Note: Interactive constraint mode only works when the software is in multi-mode multi-corner timing analysis mode.

get_library_set

```
get_library_set
     {-timing libSetName | -si libSetName}
```

Returns a Tcl list of the timing or cdB libraries for the specified library set.

Use this command after creating at least one library set (`create_library_set`).

Parameters

`-si libSetName`	Returns the cdb library names for the specified library set.
`-timing libSetName`	Returns the timing library names for the specified library set.

Examples

- The following commands create the library sets `IsCOM-1V`:

  ```
  create_library_set -name IsCOM-1V -timing [list stdcell_F_1V.lib ram_F.lib
  pad.lib]
                -si [list stdcell_F_2.cdb ram_F.cdb pad.cdb}]
  ```

- The following command returns a TCL list of the timing libraries for library set `IsCOM-1V`:

  ```
  get_library_set -timing IsCOM-1V
  ```

  The software returns the following results:

  ```
  {stdcell_F_1V.lib ram_F.lib pad.lib}
  ```

get_op_cond

```
get_op_cond
     virtualOpcondName
     {-P | -V | -T}
```

Returns the process, voltage, or temperature value for the specified operating condition.

Use this command after creating at least one operating condition (`create_op_cond`).

Parameters

`-P`	Returns the process value for the operating condition.
`-T`	Returns the temperature value for the operating condition.
`-V`	Returns the voltage value for the operating condition.
`virtualOpcondName`	Specifies the name of the operating condition to query

Examples

- The following command creates an operating condition called `myPVT2` for the library `IsCOM-2V.lib`:

```
create_op_cond -name myPVT2
   -library_file IsCOM-2V.lib
   -P 1.0
   -V 1.2
   -T 20
```

- The following command creates a collection called myPVT2-temp that contains the temperature value of the `myPVT2` operating condition:

```
set myPVT2-temp [get_op_cond myPVT2 -T]
```

get_propagated_clock

```
get_propagated_clock
     [-clock clock_list] [-pin pin_list]
     [> filename]
```

Returns the clock propagation mode, either `ideal` or `propagated`, for the specified clocks or pins.

Parameters

`> filename`	Specifies the name of a file in which to save the report. You can add the `.gz` extension to the file name to generate a compressed report. **Note:** The file name must be the last argument in the list.
`-clock clock_list`	Specifies one or more clock waveforms for which you want to retrieve the clock propagation mode information.
`-pin pin_list`	Specifies one or more pins or ports for which you want to retrieve the clock propagation mode information. You cannot specify hierarchical pins.

Example

- The following command gets the clock propagation mode information for all clock waveforms:

  ```
  get_propagated_clock -clock {*}
  ```

Related Information

`set_propagated_clock`

get_rc_corner

```
get_rc_corner
     rcCornerName
     -rcCornerAttribute
```

Returns attribute information for the specified RC corner object.

Use this command after creating at least one RC corner object (`create_rc_corner`).

Parameters

`-rcCornerAttribute`

Returns the setting for the specified RC corner attribute.

You can query one of the following attributes: `-cap_table`, `-T`, `-default_res_factor`, -detailed_res_factor, `-default_cap_factor`, `-detailed_cap_factor`, `-xcap_factor`, `-qx_lib_file`, `-qx_tech_file`, `-qx_conf_file`.

For attribute information, see the `create_rc_corner` command.

`rcCornerName` Specifies the name of the RC corner to query.

Examples

- The following commands create the RC corners `rc-cbest` and `rc-cworst`:

  ```
  create_rc_corner -name rc-cbest -cap_table myTech_bc.CapTbl -T 50
  create_rc_corner -name rc-cworst -cap_table myTech_wc.CapTbl
  ```

- The following command returns the capacitance table specified for `rc-cworst`:

  ```
  get_rc_corner rc-cworst -cap_table
  ```

  The software returns the following results:

  ```
  myTech_wc.CapTbl
  ```

get_time_unit

```
get_time_unit
```

Returns the current time unit for the session.

Example

- The following command returns the time unit for the session:

```
get_time_unit
1.000000ns
```

Related Information

`setLibraryUnit`

getAnalysisMode

```
getAnalysisMode
     [-analysisType]
     [-asyncChecks]
     [-caseAnalysis]
     [-checkType]
     [-clkNetsMarking]
     [-clkSrcPath]
     [-clockGatingCheck]
     [-clockPropagation]
     [-cppr]
     [-enableMultipleDriveNet]
     [-honorActiveLogicView]
     [-honorClockDomains]
     [-log]
     [-propSlew]
     [-quiet]
     [-sequentialConstProp]
     [-skew]
     [-timeBorrowing]
     [-timingEngine]
     [-timingSelfLoopsNoSkew]
     [-usefulSkew]
     [-useOutputPinCap]
     [-warn]
```

Returns the following information about a specified timing analysis mode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by the user

If you do not specify a parameter, the software returns information for all timing analysis mode parameters.

Parameters

parameter_names	Returns information for the specified parameters. You can specify one or more parameters. See setAnalysisMode for descriptions of the timing analysis mode parameters you can specify.

`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setAnalysisMode` parameters in Tcl list format.

Examples

- The following command returns the current setting of the `-checkType` parameter:

  ```
  getAnalysisMode -checkType
  ```

  The software returns the following information:

  ```
  -checkType setup                        # enums={setup hold}, default=setup
  setup
  ```

- The following command returns the current setting of the `-checkType` parameter in Tcl list format:

  ```
  getAnalysisMode -checkType -quiet
  ```

  The software returns the following information:

  ```
  setup
  ```

- The following command returns the current settings for the `-checkType`, `-skew`, and `-analysisType` parameters:

  ```
  getAnalysisMode -checkType -skew -analysisType
  ```

  The software returns the following information:

  ```
  -checkType setup                         # enums={setup hold}, default=setup
  -skew true                               # bool, default=true
  -analysisType bcwc                      # enums={single bcwc onChipVariation},
  default=bcwc
  {checkType setup} {skew true} {analysisType bcwc}
  ```

- The following command returns the current settings for the `-checkType`, `-skew`, and `-analysisType` parameters in Tcl list format:

  ```
  getAnalysisMode -checkType -skew -analysisType -quiet
  ```

  The software returns the following information:

  ```
  {checkType setup} {skew true} {analysisType bcwc}
  ```

getOpCond

```
getOpCond
     [–max]
     [–min]
     [-verbose]
```

Returns the operating temperature, process, or voltage conditions for the design set with the `setOpCond` command.

Parameters

`-max`	Returns the name of the maximum library operating condition.
`-min`	Returns the name of the minimum library operating condition.
`-verbose`	Controls the amount of information written to the log file.

Command Order

Use this command after importing the design and reading the timing libraries.

Examples

- The following command returns the operating temperature, process, or voltage conditions for the design:

```
getOpCond
min: BCCOM
max: WCCOM
```

- The following command returns the name of the minimum library for the design:

```
getOpCond -min
min: BCCOM
```

- The following command returns the name of the maximum library for the design:

```
getOpCond -max
max: WCCOM
```

- The following command returns the operating conditions and library names for the minimum and maximum libraries in the design:

```
getOpCond -v
min: BCCOM proc: 1.0 volt: 3.3 temp: 100
max: WCCOM proc: 2.0 volt: 3.3 temp: 120
```

- The following command returns the operating conditions and library name for the minimum library in the design:

```
getOpCond -min -v
min: BCCOM proc: 1.0 volt: 3.3 temp: 100
```

- The following command returns the operating conditions and library name for the maximum library in the design:

```
getOpCond -max -v
max: WCCOM proc: 2.0 volt: 3.3 temp: 120
```

getTimeLibFile

`getTimeLibFile` *`libName`*

Reports the library files that match the specified library name.

Parameters

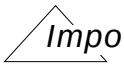
`libName`	Specifies the library name for which to check.

loadTimingCon

```
loadTimingCon
     [-incr]
     [-ilm]
     fileName
```

Loads a timing constraints file. The timing constraints file can be in any format including PrimeTime (PT) and Synopsys Design Constraints (SDC).

Note: You can also specify a list of constraints files in the configuration file.



You cannot use this command when the software is in multi-mode multi-corner mode. Instead, use the `create constraint mode` command to control the use of constraint files.

Parameters

fileName	Specifies the timing constraints file.
-incr	Adds the *fileName* information to the previously loaded constraint information. You use this command to incrementally load constraint information in an Encounter session.
-ilm	**Note:** The `-ilm` parameter is obsolete, because the command detects ILMs by default. Specifies the interface logic model (ILM) constraints file. When you use this parameter, the software reads the specified ILM constraint file, and uses it in the ILM view. This file is not used in the blackbox view.

Command Order

Use this command after importing the design and loading the timing library.

Example

The following command loads the timing constraints file `mydesign.tc`:

```
loadTimingCon mydesign.tc
```

Related Topics

- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*

read_locvlib

```
read_locvlib
     [-view view_name]
     list_of_file_names
```

Reads the specified LOCV library files into the design.

An LOCV library file contains one or more LOCV tables, which are used to derive the LOCV derating factors for timing analysis. When you specify the `select_locv_table` command, the software looks in the LOCV library file for an LOCV table that matches your specifications, and derives the derating factors from that table. An LOCV library can be a one-dimensional or two-dimensional table.

You can specify encrypted (binary) LOCV library files.

Parameters

list_of_file_names

Specifies the name of the LOCV library file to read. You can specify one or mile file names.

`-view` *view_name*

Reads the LOCV library file and uses it for the specified analysis view only.

You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

LOCV Table Format

LOCV tables are used for deriving the correct LOCV derating factors for timing analysis. An LOCV table can be defined for all cells, all nets, all cells and nets, or for a specific cell. When you specify the `select_locv_table` command, the software looks in the LOCV library file for an LOCV table that matches your specifications, and derives the derating factors from that table.

You can define an LOCV table for all cells, and tables for specific cells in the same LOCV library file. The derating factors from the tables for specific cells are only applied to those cells, and the derating factors from the all cells table is applied to the rest of the cells.

The following format is used to define an LOCV table:

```
[Apply-Cell | Apply-Net | Apply-All]
{MAX-Setup | MIN-Setup | MAX-Hold | MIN-Hold}[-{Early | Late}][-{Rise | Fall}]
```

```
[Voltage volt]
[Cell {cell_name | cell_name* | *}]
[Distance    f1  f2 ...    fx
 Stage       n    n+1 ...   n
         val11  val12 ...  val1n
         val21  val22 ...  val12n
         ...
         valx1  valx2 ...  valxn |

 Stage       n    n+1 ...   n
          val1   val2 ...   valn |

 Distance    f1   f2 ... fx
           val1 val2 ... valn
]
```

Where:

`Apply-Cell | Apply-Net | Apply-All`

Specifies whether the table is for a cell, a net, or both.

Default: `Apply-All`

`Voltage volt`

Specifies a voltage for the table. You can specify a voltage for either a cell table or a net table.

The voltage statement enables you to specify different LOCV tables for specific cells (or all cells in a library) for different power domains.

If you do not specify a voltage, the software uses the table as a global table for any voltage, if there is no table for that voltage value.

`Cell {cell_name | cell_name* | *}`

Specifies the name of a cell with which to associate the table. If you define a table for a specific cell, the software applies the derating factors from that table only to that cell. If you do not specify a cell name, the software uses the table for all cells.

Note: Do not specify a cell name in a net table (`Apply-Net`). If you do, the software generates an error.

`{MIN-Setup | MAX-Setup | MIN-Hold | MAX-Hold}[-{Early | Late}] [-{Rise | Fall}]`

Specifies the analysis type combination to associate with the table. An LOCV table can have one of several combinations:

- `MAX` (worst) and `MIN` (best)
- `Setup` and `Hold`
- `Early` and `Late`
- `Rise` and `Fall`

Use the `select_locv_table` command to specify whether to use the `MAX` or `MIN` table for setup or hold analysis. The software chooses the `Setup`/`Hold`, `Early`/`Late`, and `Rise`/`Fall` combinations.

`Distance` *`f1 f2 ... fx`*
`val1 val12 ...val1n`

Specifies an index of distance values, in microns. The values must be specified in increasing order.

The distance for a path is defined as the diagonal length of the bounding box of the path which covers:

- The output pin of the clock branch point
- All pins of cells in the clock path and data path after the clock branch point (including the end point).

The bounding box does not include the net area.

Distance definitions account for the systematic variations in the cell's timing (that is, the variations in the cell's timing due to P,V, and T with distance). The rows in the LOCV table correspond to the distance values.

`Stage` *`n n+1 ...n`*

Specifies an index of stage count integer values, in increasing order.

A cell stage count value is defined as follows:

- For a data path, it is the number of cells from the clock branch point to the end point for the data path.
- For a clock path, it is the number of cells from the clock branch point to the clock pin of the end point.

Similarly, a net stage count value is defined as follows:

- For a data path, it is the number of nets from the clock branch point to the end point for the data path.
- For a clock path, it is the number of nets from the clock branch point to the clock pin of the end point.

The clock branch point and the end point should not be counted in the cell count.

Stage count accounts for the random variation in the cell's timing. The columns in the LOCV table always correspond to the stage count values.

The following example shows an LOCV table as it is defined in the LOCV library:

```
MAX-Setup
Stage    0     1     2     3     4     5     6     7     8     9    10
Distance 0 1000 2000 4000 8000
     0.828 0.828 0.872 0.892 0.904 0.913 0.919 0.925 0.927 0.930 0.933
     0.828 0.828 0.872 0.892 0.904 0.913 0.919 0.924 0.927 0.930 0.933
     0.823 0.823 0.864 0.885 0.896 0.904 0.909 0.913 0.916 0.919 0.921
     0.813 0.813 0.853 0.869 0.879 0.885 0.889 0.892 0.895 0.896 0.899
     0.794 0.794 0.834 0.844 0.854 0.858 0.862 0.863 0.866 0.867 0.869
```

read_sdf

```
read_sdf
     [ -sdf_field {min | typ | max}
     | [-early_sdf_field {min | typ | max}]
       [-late_sdf_field {min | typ | max}]
     ]
     [-continue_on_error]
     [-ilm_filter]
     [-strict_cond_matching]
     [-scale float]
     [-view viewName]
     [-path instanceName]
     [-clock_mesh [-restructure_only]]
     [-persistent]
     [-overwrite_incremental_delay]
     sdf_filename
```

Reads an OVI Standard Delay Format (SDF) file and annotates user-specified delay information from the SDF file into the timing system. You can use the `read_sdf` command to supply precalculated delays and timing check values from an external delay calculator, or from a previous Encounter session.

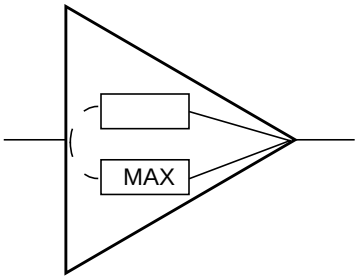
The `read_sdf` command can read version 2.1 and 3.0 of the SDF files, and can read both plain-text and GNU zipped (`.gz`) format files. The command and the Encounter software do not support the Timing Environment section of the SDF specification.

The SDF file can provide one, two, or three different values for each timing arc delay or timing check value. These values are passed in the min, typ, or max SDF delay fields, represented as min:typ:max.

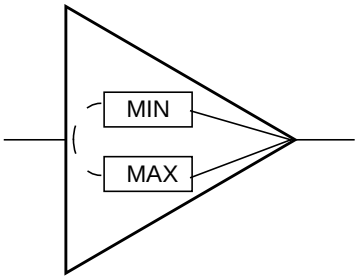
The Encounter software provides multiple delay place holders to store the calculated or annotated delay information:

- In single-corner (`setAnalysisMode -analysisType single`) analysis mode, only max delays are available.
- In best-case worst-case (`setAnalysisMode -analysisType bcwc`) mode, a min delay slot is provided for storing min delays used for hold analysis. A max delay slot stores delays used for setup analysis.
- In on-chip variation (`setAnalysisMode -analysisType onChipVariation`) mode, the early delay slots stores delays used for calculated early arrival times. The late delay slot stores delays used for calculated late arrival times.
- In multi-corner mode, an early and a late delay slot are provided for each multi-mode multi-corner analysis view.

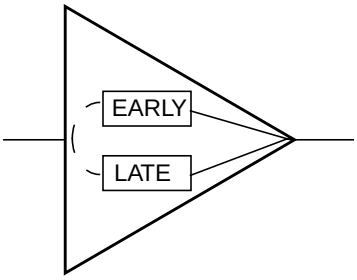
The `read_sdf` command provides parameters for controlling how the SDF min:typ:max triplet values are mapped into the Encounter software's delay slots. The correct mapping depends on how the delay information in the SDF file was created, and what type of analysis you are performing.



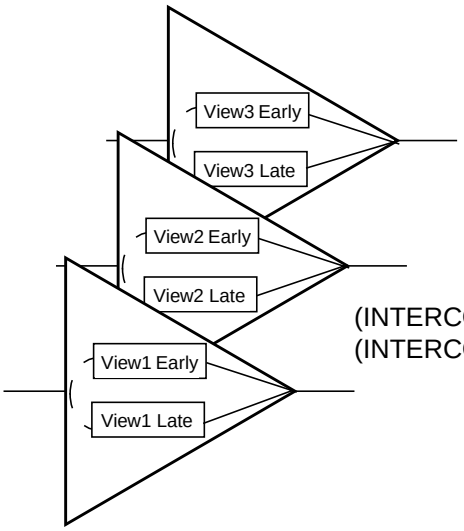
Delay slots in single-corner mode



Delay slots in best-case worst-case mode



Delay slots in on-chip variation mode



Delay slots in multi-corner mode

```
                                    min : typ : max     min : typ : max
(INTERCONNECT C1/Y RS/CK (0.123:0.456:0.789)(0.123:0.456:0.789))
(INTERCONNECT C1/Y RT/CK (0.123:0.456:0.789)(0.123:0.456:0.789))
```

Rising delays Falling delays

Standard SDF delay triplet

The timing system provides special handling for annotating incremental delays. These delays typically represent plus or minus offsets due to crosstalk effects. Incremental delay annotation to the Encounter software can be done by:

- Importing an incremental SDF file (SDF has an internal `INCREMENT` keyword) using the `read_sdf` command.
- Tcl sourcing a file of `set_annotated_delay -net -delta_only` constraints.

The incremental delay annotations can be viewed in the `report_timing` report by adding the `incr_delay` option to the `report_timing -format` list, or to the `report_timing_format` global variable.

Parameters

`-clock_mesh`

Reduces the clock mesh using the same driver-to-receiver pairs as are present in the SDF file.

By default, when the software reads a netlist with an instantiated clock mesh network, it automatically reduces it to a single driver to many receivers net. This reduction might not be consistent with similar processes in external delay calculators. An externally-generated SDF file with pre-characterized clock mesh delays might not annotate properly onto the CTE reduced network version of the clock mesh.

Note: The Encounter save/restore design process does not archive either the restructuring or delay annotations of the clock mesh. You must respecify `read_sdf -clock_mesh` after restoring a design.

`-continue_on_error`

Continues to read the file if an error occurs. By default, when an error occurs, the `read_sdf` command stops reading the file. Use this parameter to attempt to continue reading the file, if possible.

Note: Use caution if you continue, because some annotations could be missing due to the errors. Use the `report_annotations` command to find missing annotations. If the file is corrupted, obtain a new SDF file.

`-early_sdf_field {min | typ | max}`

Annotates the early delay from the specified SDF field.

In single-corner mode, all setup and hold analysis uses max delays. The software ignores any min/early delay information.

In best-case worst-case mode, use this parameter to specify which SDF field to use for hold analysis.

In on-chip variation mode, the software uses the early delay for calculation of fast paths.

`-ilm_filter` Ignores ILM-related errors. By default, the `read_sdf` command does not ignore errors reported due to missing arcs in core ILM.

`-late_sdf_field {min | typ | max}`

Annotates the late delay from the specified SDF field.

In single-corner mode, use this parameter to specify which of the SDF triplet values to annotate for both setup and hold checks.

In best-case worst-case mode, use this parameter to specify which SDF field to use for setup analysis.

In on-chip variation mode, the software uses the late delay for the calculation of slow paths.

`-overwrite_incremental_delay`

When annotating an incremental SDF file, replaces any pre-existing delta delays with the new SDF values. The total delay will be equivalent to the original base delay, plus the new delta delay.

If the delays in the incremental SDF file represent crosstalk delay push-outs from a signal integrity analysis, use the `-overwrite_incremental_delay` parameter to ensure that any stale data from a previous analysis is removed.

By default, when an incremental SDF file (SDF has an internal `INCREMENT` keyword) is annotated, the delays are added to the base delay, and also stored separately as delta delays. If an additional incremental SDF file is annotated, the software adds the existing base and delta delays with the new values.

`-path` *`instanceName`* Specifies the instance in the design to which the specified SDF file should be applied.

`-persistent` Keeps the delays from certain SDF files persistently in use during various steps of the timing optimization flow. Use this parameter to support pre-timed structures, such as clock meshes.

`-restructure_only` Restructures the delays, but does not annotate them.

Note: The `-restructure_only` parameter can only be used with the `-clock_mesh` parameter.

Default: Restructures the delays and annotates them.

`-scale` *`float`* Scales all values in the SDF file by the given number.

`-sdf_field {min | typ | max}`

Annotates the delay from the specified SDF slot.

In single-corner and best-case worst-case modes, the software uses this value for both setup and hold analysis.

In on-chip variation mode, the software uses the same value from the specified SDF slot for both the early and late delays of the specified arc.

`sdf_filename` Specifies the name of the SDF file that has the back annotation data.

`-strict_cond_matching`

Annotates only the conditions that are an exact syntax-to-syntax match between the SDF file and the `.lib` file.

Default: Uses looser conditional matching rules between the `.lib` and SDF arc specifications.

`-view` *`viewName`* Annotates the specified analysis view with the delays from the SDF file. You can use this parameter only when you are in multi-mode multi-corner analysis mode.

Examples

- The following command annotates delay information from the SDF min field in to the min/early slots, and maps the SDF max field data to the max/late delay slots:

  ```
  read_sdf sdf_filename
  ```

  Given the following SDF, the default annotation would be:

  ```
  (INTERCONNECT C1/Y RS/CK (0.123:0.456:0.789) (0.123:0.456:0.789))
  ```

  - In single-corner analysis mode, a delay of 0.789 ns is annotated to the max delay slot.
  - In best-case worst-case mode, a delay of 0.123 ns is annotated to the min delay slot, and a value of 0.789 ns is placed in the max delay slot.

  - In on-chip variation mode, a delay of 0.123 is used for early delays, and a value of 0.789 ns is used for late delays.

- The following command annotates the SDF typ field to both the early and late delay slots:

  ```
  read_sdf -sdf_field typ sdf_filename
  ```

- The following command annotates the SDF typ field to the early delay slot, and the SDF max field to the late delay slot:

  ```
  read_sdf -early_sdf_field typ -late_sdf_field max sdf_filename
  ```

- The following command annotates the SDF typ field to the late delay slot. The early delay slot is annotated with the default min SDF field:

  ```
  read_sdf -late_sdf_field typ sdf_filename
  ```

read_spdf

```
read_spdf
     SPDF_file_name
```

Reads the statistical parameter distribution format (SPDF) file. An SPDF file contains statistics of variation of process parameters. The SPDF file is required for statistical static timing analysis (SSTA).

For more information on the format of SPDF file, see Statistical Parameter Distribution Format (SPDF) File in the Encounter User Guide.

Parameters

SPDF_file_name	Specifies the name of the SPDF file.

read_twf

```
read_twf
     [-help]
     [filenames]
     [-prefix string]
     [-reset]
     [-scope string]
     [-skipconst]
     [-skiptw]
     [-strip_prefix string]
```

Reads the external Timing Window File (TWF) for noise and power calculation in CPE and ETS SI.

Parameters

filenames	Specifies the names of the timing window files.
-prefix *string*	Adds a prefix to the net or pin name.
-reset	Clears the previously read TWF files.
-scope *string*	Specifies the scope for the given hierarchichal timing window file.
-skipconst	Skips constant processing.
-skiptw	Skips timing windows processing.
-strip_prefix	Removes hierarchical path name.

report_analysis_coverage

```
report_analysis_coverage
     [-pins pin_list]
     [-check_type check_type_list]
     [-verbose check_status_list]
     [-sort {pin | refpin | checktype | slack | reason}]
     [-exclude_untested reason]
     [-view view_name]
     [{> | >> } file_name | -tcl_list]
```

Provides information about the timing checks in the design. Valid check types are clock_gating_hold, clock_gating_pulse_width, clock_gating_setup, clock_period, clock_separation, external_delay, hold, no_change_hold, no_change_setup, path_delay, pulse_width, recovery, removal, setup, skew, and time_borrow. Valid status types are: met, untested, and violated.

For each type of timing check, the command reports the number of checks that meet constraints, violated constraints, or that are untested in the `SUMMARY` section. The `DETAILS` section includes information about the signal pin, reference pin, check type, slack, and the reason for being untested.

A `met` check shows as a positive value in the `Slack` column. A `violated` check shows as a negative value in the `Slack` column. An `untested` check shows as `UNTESTED` in the `Slack` column and the reason is given in the `Reason` column. The `Reason` column is blank when a `Slack` number is given because the check has been tested.

Note: If there are no timing checks in the design of the type specified by `-check_type`, the report shows “No timing checks found.”

Tip

Use `check_timing` to look for missing constraints.

Parameters

`{> | >>}` *`file_name`*
: When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it.

  When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

  The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

  Note: To write a compressed report, add the `.gz` extension to the file name.

  Default: Report is displayed on standard output without being saved.

`-check_type` *`check_type_list`*
: Limits the report to the type of check(s) in the *`check_type_list`*.

  Valid check types are: `setup`, `hold`, `pulse_width`, `clock_period`, `clock_gating_setup`, `clock_gating_hold`, `clock_gating_pulse_width`, `recovery`, `removal`, `clock_separation`, `skew`, `no_change_setup`, `no_change_hold`, `time_borrow`, `external_delay`, and `path_delay`.

  Default: All `check_types` are reported.

`-exclude_untested {`*`untested_reason_list`*`}`
: Excludes from the report endpoints that are unconstrained due to a specific reason.

  Valid reasons are: `const`, `no_startpoint_clock`, `no_endpoint_clock`, `user_disable`, and `unknown` (which includes all other reasons, such as a false path, or no path).

`-pins` *`pin_list`*
: Limits the report to timing checks on pins in the *`pin_list`*.

`-sort {pin | refpin | checktype | slack | reason}`

Sorts the checks by the specified method while reporting detailed information using the `-verbose` option. The `-sort` option does not change the order of the columns.

The criteria by which sorting is attempted for the `slack`, `pin`, and `checktype` arguments is as follows:

- When sorting by slack, if the slack values are the same, the software then attempts to sort the checks by pin name. If the pin names are the same, the software attempts to sort by timing check type.

  When you sort by `slack`, the slack value is reported in order of least negative to most negative (worst slack last).

- When sorting by pin name, if the pin names are the same, the software then attempts to sort the checks by slack value. If the slack values are the same, the software attempts to sort by timing check type.

- When sorting by timing check type, if the check types are the same, the software then attempts to sort by slack value. If the slack values are the same, the software attempts to sort by pin name.

Default: By default, checks are sorted by slack value (`slack`).

 *Tip*

Improve usability of the report for debugging purposes by using the `-verbose` and `-sort` options as shown in these examples:

```
report_analysis_coverage -verbose untested -sort
reason
report_analysis_coverage -verbose violated -sort
slack
```

`-tcl_list`

Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation.

The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

`-verbose` *`check_status_list`*

Limits the detailed information to only the checks with the status specified.

Valid status types are: `met`, `violated`, and `untested.`

Tip

Reduce the number of lines in the `DETAILS` section and improve usability of the report by using the `-verbose` option to filter out the `met` checks as shown in this example:

```
report_analysis_coverage -verbose {violated
untested}
```

`-view` *`view_name`* Generates a timing check report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- The following command generates the following report about the timing checks in the design:

  ```
  report_analysis_coverage
  ```

TIMING CHECK COVERAGE SUMMARY

Check Type	No. of Checks	Met	Violated	Untested
Clock Gating Hold	2	0 (0%)	2 (100%)	0 (0%)
Clock Gating Setup	2	2 (100%)	0 (0%)	0 (0%)
ExternalDelay (Early)	4	4 (100%)	0 (0%)	0 (0%)
ExternalDelay (Late)	4	4 (100%)	0 (0%)	0 (0%)
Hold	16	4 (25%)	2 (12%)	10 (62%)
PulseWidth	16	14 (87%)	0 (0%)	2 (12%)
Setup	16	6 (37%)	0 (0%)	10 (62%)

TIMING CHECK COVERAGE DETAILS

Pin	Reference Pin	Check Type	Slack	Reason
BLK/BR1/CK ^	BLK/BR1/CK v	PulseWidth	UNTESTED	const
BLK/BR1/CK v	BLK/BR1/CK ^	PulseWidth	UNTESTED	const
BLK/BR1/D	BLK/BR1/CK ^	Hold	UNTESTED	const
BLK/BR1/D	BLK/BR1/CK ^	Setup	UNTESTED	const
BLK/BR2/CK ^	BLK/BR2/CK v	PulseWidth	4.854	
BLK/BR2/CK v	BLK/BR2/CK ^	PulseWidth	3.972	
BLK/BR2/D	BLK/BR2/CK ^	Setup	UNTESTED	disable
BLK/BR2/D	BLK/BR2/CK ^	Hold	UNTESTED	disable
CG1/A ^	CG1/B v	Clock Gating Hold	-5.041	
CG1/A ^	CG1/B ^	Clock Gating Setup	9.943	
CG1/A v	CG1/B v	Clock Gating Hold	-5.072	
CG1/A v	CG1/B ^	Clock Gating Setup	9.979	
CGR/CK ^	CGR/CK v	PulseWidth	4.610	
CGR/CK v	CGR/CK ^	PulseWidth	4.360	
CGR/D	CGR/CK ^	Hold	UNTESTED	No data signal
CGR/D	CGR/CK ^	Setup	UNTESTED	No data signal

- The following command filters from the report all endpoints that are unconstrained because of the reason `const`:

```
report_analysis_coverage -exclude_untested const
```

```
+-------------------------------------------------------------------+
|                  TIMING CHECK COVERAGE SUMMARY                    |
|-------------------------------------------------------------------|
|      Check Type       | No. of |   Met    | Violated | Untested |
|                       | Checks |          |          |          |
|-----------------------+--------+----------+----------+----------|
| Clock Gating Hold     | 2      | 0 (0%)   | 2 (100%) | 0 (0%)   |
| Clock Gating Setup    | 2      | 2 (100%) | 0 (0%)   | 0 (0%)   |
| ExternalDelay (Early) | 4      | 4 (100%) | 0 (0%)   | 0 (0%)   |
| ExternalDelay (Late)  | 4      | 4 (100%) | 0 (0%)   | 0 (0%)   |
| Hold                  | 16     | 4 (25%)  | 2 (12%)  | 10 (62%) |
| PulseWidth            | 16     | 14 (87%) | 0 (0%)   | 2 (12%)  |
| Setup                 | 16     | 6 (37%)  | 0 (0%)   | 10 (62%) |
+-------------------------------------------------------------------+

+-------------------------------------------------------------------------------------+
|                            TIMING CHECK COVERAGE DETAILS                            |
|-------------------------------------------------------------------------------------|
|     Pin      |  Reference   |      Check Type       |  Slack   |    Reason      |
|              |     Pin      |                       |          |                |
|--------------+--------------+-----------------------+----------+----------------|
| BLK/BR2/CK ^ | BLK/BR2/CK v | PulseWidth            |    4.854 |                |
| BLK/BR2/CK v | BLK/BR2/CK ^ | PulseWidth            |    3.972 |                |
| BLK/BR2/D    | BLK/BR2/CK ^ | Hold                  | UNTESTED | disable        |
| BLK/BR2/D    | BLK/BR2/CK ^ | Setup                 | UNTESTED | disable        |
| CG1/A ^      | CG1/B v      | Clock Gating Hold     |   -5.041 |                |
| CG1/A ^      | CG1/B ^      | Clock Gating Setup    |    9.943 |                |
| CG1/A v      | CG1/B v      | Clock Gating Hold     |   -5.072 |                |
| CG1/A v      | CG1/B ^      | Clock Gating Setup    |    9.979 |                |
| CGR/CK ^     | CGR/CK v     | PulseWidth            |    4.610 |                |
| CGR/CK v     | CGR/CK ^     | PulseWidth            |    4.360 |                |
| CGR/D        | CGR/CK ^     | Hold                  | UNTESTED | No data signal |
| CGR/D        | CGR/CK ^     | Setup                 | UNTESTED | No data signal |
```

- When the software is in multi-mode multi-corner timing analysis mode, You can specify the `-view` parameter to generate a timing check report for the specified analysis view only:

```
report_analysis_coverage -view s1
```

```
+-------------------------------------------------------------------+
|       TIMING CHECK COVERAGE SUMMARY (Analysis View:  s1)          |
|-------------------------------------------------------------------|
|      Check Type       | No. of |   Met    | Violated | Untested |
|                       | Checks |          |          |          |
|-----------------------+--------+----------+----------+----------|
| ExternalDelay (Early) | 1      | 1 (100%) | 0 (0%)   | 0 (0%)   |
| ExternalDelay (Late)  | 1      | 0 (0%)   | 1 (100%) | 0 (0%)   |
| Hold                  | 2      | 1 (50%)  | 1 (50%)  | 0 (0%)   |
| PulseWidth            | 4      | 4 (100%) | 0 (0%)   | 0 (0%)   |
| Setup                 | 2      | 1 (50%)  | 1 (50%)  | 0 (0%)   |
+-------------------------------------------------------------------+
```

Related Information

check_timing

report_timing

report_analysis_views

```
report_analysis_views
     [ -type {all | setup | hold | active} ]
```

Generates a hierarchical report of the current multi-mode multi-corner configuration. By default, the `report_analysis_views` command generates a report of all defined views in the design. Use the `-type` parameter to generate a report on a specific type of analysis view.

Use this command after creating analysis views (`create_analysis_view`), and setting active setup and hold views (`set_analysis_view`) for the design.

Parameters

`-type`	Specifies the type of report to generate. *Default:* `all`	
	`active`	Generates a report of all active setup and hold views in the design.
	`all`	Generates a report of all defined views in the design, including those that are currently inactive.
	`hold`	Generates a report of all active hold views in the design.
	`setup`	Generates a report of all active setup views in the design.

Examples

- The following command generates a hierarchical report of the active setup analysis views in the design:

  ```
  report_analysis_views -type setup
  ```

  The software reports the following results:

  ```
  Multi-Mode Analysis View Report
  ___________________________________

  + SETUP Views
     |
     + Analysis View:  io+typ-rcMax
       |
  ```

```
      + Delay Calc Corner: typ-rcMax
      |   |
      |   + library_set: libs-typ
      |   |    |
      |   |    + timing: ./lib/TSMC18_6LM/synopsys/typical.lib
      |   |    |
      |   |    + si: ./lib/TSMC18_6LM/cdb/artisan-sc-x_2001q4v0#tsmc-
T018LOSP001_19#TT#1.8V#25C.cdB
      |   |
      |   + opcond: 1.80V_25C
      |   |    |
      |   |    + library: ./lib/TSMC18_6LM/synopsys/typical.lib
      |   |    |
      |   |    + P: 1
      |   |    |
      |   |    + V: 1.8
      |   |    |
      |   |    + T: 25
      |   |
      |   + opcond_library: ./lib/TSMC18_6LM/synopsys/typical.lib
      |   |
      |   + rc_corner: rcMax
      |   |    |
      |   |    + cap_table: ./lib/TSMC18_6LM/techfiles_6lm/t018losp001_6ml-
rcmax.capTbl
      |   |    |
      |   |    + default_res_factor: 1.12
      |   |    |
      |   |    + detailed_res_factor 1.12
      |   |    |
      |   |    + default_cap_factor: 0.88
      |   |    |
      |   |    + detailed_cap_factor: 0.76
      |   |    |
      |   |    + xcap_factor: 1
      |
      + Constraint Mode: io
      |   |
      |   + SDC Constraint Files: ./data/constraints/tdsp_core-io.sdc
   |
   + Analysis View:  core+typ-rcMax
```

```
      |
      + Delay Calc Corner: typ-rcMax
      |   |
      |   + library_set: libs-typ
      |   |    |
      |   |    + timing: ./lib/TSMC18_6LM/synopsys/typical.lib
      |   |    |
      |   |    + si: ./lib/TSMC18_6LM/cdb/artisan-sc-x_2001q4v0#tsmc-
T018LOSP001_19#TT#1.8V#25C.cdB
      |   |
      |   + opcond: 1.80V_25C
      |   |    |
      |   |    + library: ./lib/TSMC18_6LM/synopsys/typical.lib
      |   |    |
      |   |    + P: 1
      |   |    |
      |   |    + V: 1.8
      |   |    |
      |   |    + T: 25
      |   |
      |   + opcond_library: ./lib/TSMC18_6LM/synopsys/typical.lib
      |   |
      |   + rc_corner: rcMax
      |   |    |
      |   |    + cap_table: ./lib/TSMC18_6LM/techfiles_6lm/t018losp001_6ml-
rcmax.capTbl
      |   |    |
      |   |    + default_res_factor: 1.12
      |   |    |
      |   |    + detailed_res_factor 1.12
      |   |    |
      |   |    + default_cap_factor: 0.88
      |   |    |
      |   |    + detailed_cap_factor: 0.76
      |   |    |
      |   |    + xcap_factor: 1
      |
      + Constraint Mode: core
      |   |
      |   + SDC Constraint Files: ./data/constraints/tdsp_core-core.sdc
|
```

report_annotated_check

```
report_annotated_check
     [-check_type check_type_list]
     [-missing_delays]
     [-missing_setup]
     [-missing_hold]
     [-missing_recovery]
     [-missing_removal]
     [-missing_no_change]
     [-missing_period]
     [-missing_skew]
     [-missing_mpw]
     [-max_missing integer]
     [-max_line number]
     [-list_annotated]
     [-list_non_annotated]
     [-view viewName]
     [> filename | -tcl_list]
```

Reports coverage of annotated timing checks.

Options and Arguments

`-check_type` *`check_type_list`*

Allows you to specify the check type.

The valid check types are `setup`, `hold`, `pulse_width`, `clock_period`, `clock_gating_setup`, `clock_gating_hold`, `clock_gating_pulse_width`, `data_setup`, `data_hold`, `recovery`, `removal`, `clock_separation`, and `skew`.

`>` *`filename`*

Writes the report to the specified file name. If the file already exists, the software overwrites it.

The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the `.gz` extension to the file name.

Default: Report is displayed on standard output without being saved.

`-list_annotated` Lists the detailed annotation information for the following timing checks:

- setup
- hold
- recovery
- removal
- pulse-width
- nochange-setup
- clock-period
- skew

`-list_non_annotated` Lists the timing checks that do not have annotated delay information. The software checks for the following timing checks:

- setup
- hold
- recovery
- removal
- pulse-width
- nochange-setup
- clock-period
- skew

`-missing_delays` Reports all the timing checks that are missing SDF annotations, in addition to the total coverage report.

Important

This parameter is obsolete and will be removed in a future release. Cadence recommends that you use `-list_non_annotated` instead.

`-missing_hold` Provides a detailed report of the missing annotations on hold timing checks.

`-max_line` *`number`*
: Controls the number of output lines reported by the `-list_annotated` or `list_non_annotated` parameter.

`-max_missing` *`integer`*
: Limits the number of missing annotations to the number specified.
Default: 1000

This parameter is obsolete and will be removed in a future release. Cadence recommends that you use `-max_line` instead.

`-missing_mpw`
: Provides a detailed report of the missing annotations on both mpwl and mpwh timing checks.

`-missing_no_change`
: Provides a detailed report of the missing annotations on the `no_change` timing checks.

`-missing_period`
: Provides a detailed report of the missing annotations on period timing checks.

`-missing_recovery`
: Provides a detailed report of the missing annotations on recovery timing checks.

`-missing_removal`
: Provides a detailed report of the missing annotations on removal timing checks.

`-missing_setup`
: Provides a detailed report of the missing annotations on setup timing checks.

`-missing_skew`
: Provides a detailed report of the missing annotations on skew timing checks.

`-tcl_list`
: Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation.

The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

`-view` *`viewName`*
: Reports coverage of annotated timing checks for the specified analysis view. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

  Default: Reports annotated timing checks for the default analysis view.

Examples

- The following command reports missing annotated setups in SDF:

  ```
  report_annotated_check -missing_setup
  ```

- The following command reports all the missing annotated timing checks in SDF:

  ```
  report_annotated_check -missing_delays
  ```

report_annotated_delay

```
report_annotated_delay
     [-missing_delays]
     [-max_missing value]
     [-ignore_tied_low_high_net_arcs]
     [-list_annotated]
     [-list_non_annotated]
     [-view viewName]
     [-max_line number]
     [> filename | -tcl_list]
```

Reports the coverage of the annotations on a design. These annotations include SDF delay arc annotations.

Use the `-missing_`delays option to get a detailed report of all delay arcs and wires that are missing annotations. You can use the information in a Tcl program.

Note: The `report_annotated_delay` command reports only the first 1000 missing delay arcs, which is the default value of the `-max_missing` option. Set this option to a higher value if needed, for example: `report_annotated_delay -max_missing 1000000`.

Options and Arguments

`> filename`

Writes the report to the specified file name. If the file already exists, the software overwrites it.

The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the `.gz` extension to the file name.

Default: Report is displayed on standard output without being saved.

`-ignore_tied_low_high_net_arcs`

Ignores timing arcs of power and ground nets.

`-list_annotated` Lists detailed annotation information for the following annotated arcs:

- annotated internal net arcs
- annotated input port arcs
- annotated output port arcs
- annotated cell arcs

`-list_non_annotated` Lists the arc information for the non-annotated arcs corresponding to the following types of arcs:

- annotated internal net arcs
- annotated input port arcs
- annotated output port arcs
- annotated cell arcs

`-max_line` *`number`*

Controls the number of output lines reported by `-list_annotated` or `-list_non_annotated` parameter.

`-max_missing` *`value`*

Limits the number of missing annotations to the number specified.
Default: 1000

This parameter is obsolete and will be removed in a future release. Cadence recommends that you use `-max_lines`.

`-missing_delays` Reports the arcs that are missing SDF annotations.
Default: Only a summary of the coverage is reported.

This parameter is obsolete and will be removed in a future release. Cadence recommends that you use `-list_non_annotated`.

`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together. See Example 32-16 on page 1728 for an example of the `-tcl_list` output.
`-view` *`viewName`*	Generates the report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- If your SDF has only cell arcs, and net delays are included in the cell delays, you can report the status of annotation for cell arcs, internal net arcs, net arcs from primary inputs, and net arcs to primary outputs, and all timing arcs separately.

  The following report is generated for `report_annotated_delay` for delay arcs types:

```
-------------------------------------------------------------------------
   Delay Arc Type       Total     Annotated      Not          Percentage
                                                Annotated     Annotated
-------------------------------------------------------------------------
            Cell arcs      551319      353103       198216         64.046951
       Input net arcs          18           7           11         38.888889
    Internal net arcs      501156      304341       196815         60.727798
      Output net arcs         597         456          141         76.381912
                    -
      All timing arcs     1053090      657907       395183         62.473957
-------------------------------------------------------------------------
```

report_annotations

```
report_annotations
     [-missing_resistances]
     [-missing_capacitances]
     [-missing_rc]
     [-missing_delays]
     [-missing_spf]
     [-max_missing integer]
     [-tcl_list]
     [-ignore_floating_nets]
     [-ignore_tied_low_high_nets]
     [-ignore_tied_low_high_net_arcs]
     [-list_annotated]
     [-list_non_annotated]
     [-view viewName]
     [> filename]
```

Reports the coverage of the annotations on a design. These annotations include SDF delay arc annotations, SPF annotations, wire resistances, and wire capacitances.

Use the `-missing_*` options to get a detailed report of all delay arcs and wires that are missing annotations. You can use the information in a Tcl program.

Note: The `report_annotations` command reports only the first 1000 nets, which is the default value of the `-max_missing` option. Set this option to a higher value if needed, for example: `report_annotations -max_missing 1000000`.

Parameters

`> filename`
: Writes the report to the specified file name. If the file already exists, the software overwrites it.

  The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

  Note: To write a compressed report, add the `.gz` extension to the file name.

  Default: Report is displayed on standard output without being saved.

`-ignore_floating_nets`
: Ignores nets without drivers, power, and ground nets.

`-ignore_tied_low_high_net_arcs`

Ignores timing arcs of power and ground nets.

`-ignore_tied_low_high_nets`

Ignores nets connected to `tied_low` and `tied_high` cells.

`-list_annotated` Lists detailed annotation information for the following annotated arcs:

- annotated internal net arcs
- annotated input port arcs
- annotated output port arcs
- annotated cell arcs

`-list_non_annotated` Lists the arc information for the non-annotated arcs corresponding to the following types of arcs:

- annotated internal net arcs
- annotated input port arcs
- annotated output port arcs
- annotated cell arcs

`-max_missing` *`integer`*

Limits the number of missing annotations to the number specified.
Default: 1000

`-missing_capacitances`

Reports the nets that are missing capacitances.
Default: Only a summary of the coverage is reported.

`-missing_delays` Reports the arcs that are missing SDF annotations.
Default: Only a summary of the coverage is reported.

`-missing_rc` Reports the nets that are missing resistances and capacitances.
Default: Only a summary of the coverage is reported.

`-missing_resistances`

Reports the nets that are missing resistances.
Default: Only a summary of the coverage is reported.

`-missing_spf`	Reports the nets that are missing SPF data. *Default*: Only a summary of the coverage is reported.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together. See Example 32-16 on page 1728 for an example of the `-tcl_list` output.
`-view` *`viewName`*	Reports the coverage of annotations for the specified analysis view. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode. *Default:* Reports against the default analysis view.

Examples

- The following command reports the nets that are missing capacitances:

  ```
  report_annotations -missing_capacitances
  ```

  Default: Only a summary of the coverage is reported.

- The following commands show the difference between the summary and the `-missing_spf` reports:

Example 32-2 report_annotations Summary Report

```
+-------------------------------------------------------------------------------------+
| Total |  Total C  | Percentage |  Total R  | Percentage | Total SPF | Percentage |
|  Nets | Annotated|     C      | Annotated |     R      | Annotated |    SPF     |
|       |          |  Annotated |           |  Annotated |           |  Annotated |
|-------+----------+------------+-----------+------------+-----------+------------|
|   21  |        0 |   0.000000 |         0 |   0.000000 |        15 |   0.714286 |
+-------------------------------------------------------------------------------------+
+---------------------------------+
| Total |   Total    | Percentage |
|  Arcs |   Arcs     |    Arcs    |
|       | Annotated |  Annotated |
|-------+-----------+------------|
|   272 |         0 |   0.000000 |
+---------------------------------+
```

Example 32-3 report_annotations -missing_spf Report

```
+-------------------------------------------------------------------------+
|           |          Min          |          Typ          |          Max        |
|-----------+-----------------------+-----------------------+---------------------|
|  Net Name |   R |   C   |  SPF  |   R   |   C   |  SPF  |   R   |   C   |  SPF  |
|-----------+-------+-------+-------+-------+-------+-------+-------+-------+-------|
|  bar/n019 |   - |   -   |   N   |   -   |   -   |   N   |   -   |   -   |   N   |
|  bar/n018 |   - |   -   |   N   |   -   |   -   |   N   |   -   |   -   |   N   |
|  bar/n017 |   - |   -   |   N   |   -   |   -   |   N   |   -   |   -   |   N   |
+-------------------------------------------------------------------------+
```

- If your SDF has only cell arcs, and net delays are included in the cell delays, you can report the status of annotation for cell arcs, internal net arcs, net arcs from primary inputs, and net arcs to primary outputs, and all timing arcs separately as shown below:

Example 32-4 report_annotations Report for Delay Arc Types

```
+-------------------------------------------------------------------+
|  Delay Arc Type   |  Total  | Annotated |    Not    | Percentage |
|                   |         |           | Annotated |  Annotated |
|-------------------+---------+-----------+-----------+------------|
|         Cell arcs |  551319 |    353103 |    198216 |  64.046951 |
|    Input net arcs |      18 |         7 |        11 |  38.888889 |
| Internal net arcs |  501156 |    304341 |    196815 |  60.727798 |
|   Output net arcs |     597 |       456 |       141 |  76.381912 |
+-------------------+---------+-----------+-----------+------------+
|   All timing arcs | 1053090 |    657907 |    395183 |  62.473957 |
+-------------------------------------------------------------------+
```

Related Information

report_net

report_case_analysis

```
report_case_analysis
     [-all]
     [-nosplit]
     [-dir]
     [-view view_name]
     [-propagated]
     [-verbose]
     [pins_and_ports_list]
     [{> | >>} filename | -tcl_list]
```

Generates a User Case Analysis table, which reports ports and pin that have a `set case analysis` constraint. These ports and pins are constant only for timing analysis. You can optionally generate a Logic Constant Value table, which reports ports and pins whose constant status is part of the design. Information from this table includes ports or pins with `set_logic` constraints (`set_logic_zero` and `set_logic_one`), netlist explicit constants, and output pins with constant output functions.

Parameters

`{> | >>} filename`	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-all`	Specifies that the Logic Constant Value table report pins or ports with `set_logic` constants, constants in the netlist, and output pins with a constant function. These output pins are from library cells or netlist modules.
`-nosplit`	Specifies that reports with long names will not be split into multiple lines.

`-dir`	Specifies that the report include the direction of the pin or port, which is reported as input, output, or inout (internal pins may not have constant constraints).
`pins_and_ports_list`	Generates a report only for the pins and ports in the specified list. *Default:* Generates a report for all pins and ports in the design.
`-propagated`	Generates a report that includes all constant pins, including pins with propagated constants. When specified, the report column header for constant values changes to `Constant value`. *Default:* Generates a report that includes only pins with a `set_case_analysis` constraint. (The column header for the constant values is labeled as `User case analysis value`.)
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together. See Example 32-16 on page 1728 for an example of the `-tcl_list` output.
`-verbose`	Generates a report in which each pin is listed on a separate line with a cause. If the pin is a propagated constant, a table is included that reports a chain of constants from an original cause to the reported pin. *Default:* All pins are reported in a table.
`-view` *`view_name`*	Generates a User Case Analysis table for the specified analysis view only. To specify this parameter, the software must be in multi-mode multi-corner (MMMC) analysis mode. *Default:* When in MMMC analysis mode, generates a table for all active analysis views

Examples

- The following command uses the `-dir` option, which shows that 1 is on the input part of the `timing_const_on_Z/Z` pin:

```
report_case_analysis -dir
```

The following report is generated:

```
+-----------------------------------------+
|        Pin name         |  Dir  |   User   |
|                         |       |   case   |
|                         |       | analysis |
|                         |       |  value   |
|-------------------------+-------+----------|
| timing_const_on_Z_out/Z | inout | 0        |
| timing_const_on_Z_in/Z  | input | 1        |
| A_case_0                | input | 0        |
| A_case_1                | input | 1        |
+-----------------------------------------+
```

- The following command generates a report that includes all constant pins, including pins with propagated constants. When specified, the report column header for constant values changes to `Constant value`.

  ```
  report_case_analysis -propagated
  ```

  The following report is generated:

  ```
  +-------------------+
  |  Pin   | Constant |
  |  name  |  value   |
  |--------+----------|
  | in     | 0        |
  | inv1/A | 0        |
  | inv1/Z | 1        |
  | out    | 1        |
  +-------------------+
  ```

- The following command generates a report in which each pin is listed on a separate line. The report includes all constant pins, including pins with propagated constants. The pin `buf2/buf1/A` is reported because of a `set_case_analysis 0` assertion on pin `in`.

  ```
  report_case_analysis -propagated -verbose  buf2/buf1/A
  ```

The following report is generated:

```
+------------------------+
|  Pin name   | Constant |
|             |  value   |
|-------------+----------|
| in          | 0        |
| buf1/buf1/A | 0        |
| buf1/buf1/Y | 0        |
| buf2/buf1/A | 0        |
+------------------------+
```

- The following report shows that there are no `set_case_analysis` constraints but there are logical constraints:

```
+-------------------------------------------------------------------+
|                      Pin name                        |  Logic     |
|                                                      | constant   |
|                                                      |  value     |
|------------------------------------------------------+------------|
| module_const_and_inout/Z                             | 1          |
| module_const_and_out/Z                               | 1          |
| module_const/Z                                       | 1          |
| u_internal_pin/A                                     | 0          |
| u1/A                                                 | 0          |
| u1/a_really_long_name_that_is_much_too_long_to_fit   | 0          |
| _in_a_normal_column_so_is_normally_split/A           |            |
| int_A_const_1/A                                      | 1          |
| int_A_const_0/A                                      | 0          |
| Dout[16]                                             | 0          |
| Dout[15]                                             | 0          |
| tie_high_t_state_const1/E                            | 1          |
| tie_high_t_state_const0/E                            | 0          |
| tieHi/Z                                              | 1          |
+-------------------------------------------------------------------+
```

- The following report shows how a long pin name is split onto two lines if you do not specify the `-nosplit` option:

```
+-------------------------------------------------------------+
|                       Pin name                              |  Logic   |
|                                                             | constant |
|                                                             |  value   |
|-------------------------------------------------------------+----------|
| u1/a_really_long_name_that_is_much_too_long_to_fit          | 0        |
| _in_a_normal_column_so_is_normally_split/A                  |          |
+-------------------------------------------------------------+----------+
```

- The following report uses the `-nosplit` option:

```
+-------------------------------------------------------------------------------------------------------+
|                                  Pin name                                                             |  Logic   |
|                                                                                                       | constant |
|                                                                                                       |  value   |
|-------------------------------------------------------------------------------------------------------+----------|
| u1/a_really_long_name_that_is_much_too_long_to_fit_in_a_normal_column_so_is_normally_split/A          | 0        |
+-------------------------------------------------------------------------------------------------------+----------+
```

- The following example shows the report generated when the software is in multi-mode multi-corner analysis mode. It lists the ports and pins with `set_case_analysis` constraints for each active analysis view.

Example 32-5

```
+-----------------------------------------------+
|  Pin name   |  User case      |  View name    |
|             | analysis value  |               |
|-------------+-----------------+---------------|
| reset       | 1               | io+typ-rcMax  |
+-----------------------------------------------+
```

Related Commands

`set_case_analysis`

report_cell_instance_timing

```
report_cell_instance_timing
     list_of_instances
     [-early | -late]
     [{> | >>} filename | -tcl_list]
```

Reports the following information about the timing context for all pin and delay arcs of the specified instances:

- Arrival time
- Required time
- Slack
- Delays

Note: Internal pins (pins inside the instance boundary) are also reported.

The `report_cell_instance_timing` command uses the results of earlier timing analysis runs; therefore, you must run the `report_timing` command before using `report_cell_instance_timing`. To analyze the setup (maximum) paths, you must use the `setAnalysisMode -checkType setup` command and run the `report_timing` command before using the `-late` parameter. Similarly, to analysis the hold (minimum) paths, you must use the `setAnalysisMode -checkType hold` command and run the `report_timing` command before using the `-early` parameter.

The `report_cell_instance_timing` timing report contains the following tables:

- The first table in the report contains information about the pins of the instance.

  The propagated slew is the slew that is propagated to the pin, based on the slew propagation mode.

- The second table contains information about the arcs through the instance.

  The slew out is the slew calculated at the arc output from the arc input slew, using the timing model.

- The third table, generated for sequential devices, contains information about the timing checks, if present, on the cell.

  Only checks that are from the library are shown in this table. Automatic checks, such as clock gating setup and hold, are not reported.

The *Phase* column of each table can contain the following symbols:

- `D` indicates it is a data signal in the clock domain.

- `C` indicates it is a clock signal in the clock domain.
- `P` indicates it is a positive phase.
- `N` indicates it is a negative phase.
- Asterisk (`*`) indicates there is a timing exception associated with the signal.

  Note: Currently, it is not possible to expand the asterisk to determine which exception is associated with the reported timing phase.

When the software is in multi-mode multi-corner timing analysis mode, the timing context information is reported for each active analysis view, with the view name listed in the *Phase* column.

Parameters

`{> | >>}` *`filename`*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-early` | `-late`	Reports the information for early (maximum) path (data hold/ clock setup) or late (minimum) path (data setup/clock hold). You must use the `setAnalysisMode -checkType hold` command and run the `report_timing` command before using the `-early` parameter. Similarly, you must use the `setAnalysisMode -checkType setup` command and run the `report_timing` command before using the `-late` parameter. *Default*: If you do not specify any of the parameters, the software uses the `-late` parameter.
`list_of_instances`	

	Specifies a flat (non-hierarchical) cell instance or list of cell instances. Use curly braces (`{}`) to enclose the list and separate the instance names with white space.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together. See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

Examples

- The following example reports information about the timing context of instance `reg01`:

Example 32-6 report_cell_instance_timing reg01 > Report

```
+-------------------------------------------------------------+
|                    Instance reg01 of LD1                    |
+-------------------------------------------------------------+
|-------------------------------------------------------------|
|  Pin  | Propagated | Arrival | Required | Slack |   Phase   |
|       |    Slew    |         |          |       |           |
|-------+------------+---------+----------+-------+-----------|
|   Q ^ |       0.09 |    0.66 |     1.78 |  1.12 | CLK(D)(P)|
|   Q v |       0.06 |    0.65 |     1.77 |  1.12 | CLK(D)(P)|
|  QN ^ |       0.02 |    0.66 |          |       | CLK(D)(P)|
|  QN v |       0.03 |    0.71 |          |       | CLK(D)(P)|
|   D ^ |       0.03 |    1.25 |     0.08 | -1.17 | CLK(D)(P)|
|   D v |       0.03 |    1.31 |     0.07 | -1.24 | CLK(D)(P)|
|   G ^ |       0.00 |    0.04 |     1.16 |  1.12 | CLK(C)(P)|
|   G v |       0.00 |    2.03 |     5.25 |  3.22 | CLK(C)(P)|
+-------------------------------------------------------------+
+-----------------------------------------------------+
|                  Instance reg01 of LD1              |
|-----------------------------------------------------|
|      Arc      |     Slew      |       |       |     |
|---------------+---------------+-------+-------+-----|
|  From |  To   |  In   |  Out  |  Load | Delay | Phase |
|-------+-------+-------+-------+-------+-------+-----|
|   D ^ |  QN v |  0.03 |  0.03 |       |  0.62 | CLK(D)(P) |
|   D v |  QN ^ |  0.03 |  0.02 |       |  0.59 | CLK(D)(P) |
|   D ^ |   Q ^ |  0.03 |  0.09 |  0.12 |  0.58 | CLK(D)(P) |
|   D v |   Q v |  0.03 |  0.07 |  0.12 |  0.57 | CLK(D)(P) |
|   G ^ |  QN ^ |  0.00 |  0.02 |       |  0.62 | CLK(C)(P) |
|   G ^ |  QN v |  0.00 |  0.03 |       |  0.67 | CLK(C)(P) |
|   G ^ |   Q ^ |  0.00 |  0.09 |  0.12 |  0.62 | CLK(C)(P) |
|   G ^ |   Q v |  0.00 |  0.06 |  0.12 |  0.61 | CLK(C)(P) |
+-----------------------------------------------------+
+-------------------------------------------------------------------------+
|                          Instance reg01 of LD1                          |
|-------------------------------------------------------------------------|
|       |     Pins      |       |       |        |       Phase           |
|-------+---------------+-------+-------+--------+-----------------------|
| Check |  Sig  |  Ref  | Slack | Delay | Adjust |    Sig    |    Ref    |
|-------+-------+-------+-------+-------+--------+-----------+-----------|
| HOLD  |   D ^ |   G v |  3.22 |  0.01 | 4.00   | CLK(D)(P) | CLK(C)(P) |
| HOLD  |   D v |   G v |  3.29 |  0.01 | 4.00   | CLK(D)(P) | CLK(C)(P) |
| SETUP |   D ^ |   G v |  0.73 |  0.05 | 0.00   | CLK(D)(P) | CLK(C)(P) |
| SETUP |   D v |   G v |  0.53 |  0.19 | 0.00   | CLK(D)(P) | CLK(C)(P) |
+-------------------------------------------------------------------------+
```

- If a constant is asserted or propagated to the instance pin, it is reported in the *Phase* column of the pin table, as shown in the following example:

Example 32-7 report_cell_instance_timing xor2 > Report

```
+-------------------------------------------------------------------------+
|                        Instance xor2 of XOR2                            |
|-------------------------------------------------------------------------|
|  Pin  |  Dir  | Propagated | Arrival | Required | Slack | Phase         |
|       |       |    Slew    |         |          |       |               |
|-------+-------+------------+---------+----------+-------+---------------|
|   Y ^ |   OUT |       1.00 |    1.00 |     0.00 | -1.00 |  clk(C)(P)    |
|   Y v |   OUT |       1.00 |    6.00 |          |       |  clk(C)(P)    |
|   A ^ |    IN |       0.00 |    0.00 |    -1.00 | -1.00 |  clk(C)(P)    |
|   A v |    IN |       0.00 |    5.00 |          |       |  clk(C)(P)    |
|     B |    IN |            |         |          |       | Constant 0    |
+-------------------------------------------------------------------------+

+-----------------------------------------------------------+
|                  Instance xor2 of XOR2                    |
|-----------------------------------------------------------|
|      Arc      |     Slew      |       |       |           |
|---------------+---------------+-------+-------+-----------|
|  From |  To   |  In   |  Out  |  Load | Delay |   Phase   |
|-------+-------+-------+-------+-------+-------+-----------|
|   A ^ |   Y ^ |  0.00 |  1.00 |  1.00 |  1.00 | clk(C)(P) |
|   A ^ |   Y v |  0.00 |       |  1.00 |       | clk(C)(P) |
|   A v |   Y ^ |  0.00 |       |  1.00 |       | clk(C)(P) |
|   A v |   Y v |  0.00 |  1.00 |  1.00 |  1.00 | clk(C)(P) |
|   B ^ |   Y ^ |       |       |  1.00 |       |           |
|   B ^ |   Y v |       |       |  1.00 |       |           |
|   B v |   Y ^ |       |       |  1.00 |       |           |
|   B v |   Y v |       |       |  1.00 |       |           |
+-----------------------------------------------------------+
```

- When the software is in multi-mode multi-corner analysis mode, the timing context information is reported for each active analysis view. The view name is reported in the *Phase* column, as shown in the following example:

```
+-------------------------------------------------------------------------+
|                          Instance U1 of BUFX2                           |
|-------------------------------------------------------------------------|
|  Pin  |  Dir  | Propagated | Arrival | Required |  Slack  | Phase        |
|       |       |    Slew    |         |          |         |              |
|-------+-------+------------+---------+----------+---------+--------------|
|   A ^ |    IN |      0.093 |  11.535 |    5.857 |  -5.677 | PH1(D)(P)(s1)|
|   A v |    IN |      0.076 |  11.302 |    5.702 |  -5.600 | PH1(D)(P)(s1)|
|   A ^ |    IN |      0.093 |  20.762 |    5.857 | -14.905 | PH1(D)(P)(s2)|
|   A v |    IN |      0.076 |  20.344 |    5.702 | -14.642 | PH1(D)(P)(s2)|
|   Y ^ |   OUT |      0.084 |  29.762 |   14.857 | -14.905 | PH1(D)(P)(s2)|
|   Y v |   OUT |      0.065 |  29.344 |   14.702 | -14.642 | PH1(D)(P)(s2)|
|   Y ^ |   OUT |      0.084 |  16.535 |   10.857 |  -5.677 | PH1(D)(P)(s1)|
|   Y v |   OUT |      0.065 |  16.302 |   10.702 |  -5.600 | PH1(D)(P)(s1)|
+-------------------------------------------------------------------------+
```

Related Information

report_net

report_timing

report_clock_gating_check

```
report_clock_gating_check
     [object_list]
     [-view view_name]
     [{> | >>} file_name]
```

Reports information of all or specific clock gating checks of the current design. If you specify a list of pins, cells, or clocks, the clock gating check information is reported for the specified objects only. If you use the `report_clock_gating_check` command without an object list, all clock gating check information is reported.

The `Type` column of the generated report (see Example on page 1668) can contain one of the following abbreviations:

- `I` indicates that the clock gating check is inferred.
- `L` indicates that the clock gating check is from the library (integrated clock gating check).

Set the `report_precision` global variable using the `set_global` command to specify the accuracy of the result in terms of number of significant digits.

Parameters

`object_list`

Specifies a list of pins, instances, or clocks. A collection of these can also be specified.

- Pin: Any pin of a clock gating cell can be specified and the check information will be reported for that pin.
- Instance: Any instance name of a clock gating cell can be specified and the check information will be reported for that instance.
- Clock: Only a clock at a given clock (reference) pin of a cell can be specified for reporting the check information. Check information about the clocks at enable (signal) pin cannot be reported.

If the *`object_list`* parameter is not specified, all the clock gating check points (inferred and library) of the current design are reported.

`-view` *`view_name`*	Generates a clock gating check point report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner (MMMC) timing analysis mode. *Default:* Generates a report for the active MMMC view.
`{> | >>}` *`filename`*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

Example

The following command returns a clock gating check report for the current design:

```
report_clock_gating_check
```

The clock gating check report returned by this command is as follows:

```
+-------------------------------------------------------------------------+
|                       Clock Gating Check Report                         |
|-------------------------------------------------------------------------|
| Instance | Enable | Clock | Type | Level | Setup | Setup | Hold | Hold |
|          |        |       |      |       |  Rise |  Fall | Rise | Fall |
|----------+--------+-------+------+-------+-------+-------+-------+----|
|    i1    |    A   |   B   |   I  |   L   | 0.000 | 0.000 |   -  |  -   |
|    i2    |    B   |   A   |   I  |   H   | 0.000 | 0.000 |   -  |  -   |
|   icg    |  CEN   |  CLK  |   L  |   H   | 0.149 | 0.181 |   -  |  -   |
+-------------------------------------------------------------------------+
```

report_clock_timing

```
report_clock_timing
     -type [ skew
           | interclock_skew
           | jitter
           | summary
           | latency
              [ [-launch | -capture]
              | [-histogram [-histogram_range {from to}]]
                [-logic_level [-source {clock_root | generated_clock}]]
              ]
           ]
     [-early | -late]
     [-clock clock_list]
     [-from_clock from_clock_list]
     [-to_clock to_clock_list]
     [-from from_list]
     [-to to_list]
     [-nworst worst_entries]
     [-greater_than lower_limit]
     [-view view_name]
     [-verbose]
     [-format column_list]
     [> filename | -tcl_list]
```

Generates a clock skew report for the current design. The software generates a separate report for each specified clock or pair of clocks. You specify the clocks using the `-clock` parameter, and clock pairs using the `-from_clock` and `-to_clock` parameters. You can use the `-nworst` and `-greater_than` parameters with the `-from_clock` and `-to_clock` parameters to generate reports with skew to the specified sink pins. You can use the `-early` and `-late` parameters to specify the type of skew to be reported.

Note: To remove common path pessimism from the reported skew, use the `setAnalysisMode -cppr both` command before using the `report_clock_timing` command. The report contains a separate column for common path adjustment values.

Parameters

`> filename`

Writes the report to the specified file name. If the file already exists, the software overwrites it.

The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the `.gz` extension to the file name.

Default: Report is displayed on standard output without being saved.

`-clock clock_list`

Specifies the list of clocks to be reported. The software generates a separate report for each clock that you specify in the *`clock_list`*.

Default: Skew report is generated for all clocks in the design.

`-early`

Uses hold skew to generate the report. Hold skew is the difference between minimum latency at the start point flip-flop and the maximum latency at the end point flip-flop.

Default: Uses the setup skew.

`-format column_list`

Specifies which columns to display for path reports, and the order in which they should appear.

For example:

```
-format {instance arc pin cell slew load}
```

The `-format` parameter can be used only for latency or skew reports (`-type skew`, `-type interclock_skew`, or `-type latency`), and if the `-verbose` parameter is specified.

See Table 32-7 on page 1757 for a list of valid arguments.

`-from from_list`

Specifies the list of sequential device clock pins or ports to consider as from-pins in the generated report. If any pin in the *`from_list`* is not a sequential device clock pin, it is replaced with all the sequential device clock pins in the transitive fanout of the named pin. If there are no sequential pins in the transitive fanout of the pin, the software drops the pin from the list and generates a warning message.

`-from_clock` *`from_clock_list`*

Specifies the list of from-clocks to be used in generating the current interclock skew report. You can use this option only in an interclock skew report.

Default: Report is generated considering all clocks as from-clocks.

`-greater_than` *`lower_limit`*

Reports skews with values greater than the *`lower_limit`* value.

`-histogram` Generates a textual histogram of the latency report.

The `-histogram` parameter can be used only when `-type latency` is specified. When specified in conjunction with the `-clock` parameter, the software limits the latency reporting to the specified clocks only. When specified with the `-greater_than` parameter, the software limits the latency reporting to clocks with a latency value that is greater than the specified value.

Note: The `-histogram` parameter cannot be specified in conjunction with the following parameters: `-launch`, `-capture`, `-from`, `-to`, `-from_clock`, `-to_clock`, `-verbose`, `-nworst`.

`-histogram_range` *`interval_size`*

Generates the histogram report at the specified intervals. The interval size specified must be a positive integer.

For example, if the maximum latency value is 437, and you specify `-histogram_range 50`, the histogram would be generated with intervals of: 0-50, 50-100, 100-150, ... 400-450.

This parameter can be used only when the `-histogram` parameter is specified.

`-late` Uses setup skew to generate the report. Setup skew is the difference between the maximum latency at the start point flip-flop and minimum latency at the end point flip-flop.

`-launch | -capture`

When specified with `-type latency`, generates a report of either the launch latency values (`-launch`), or the capture latency values (`-capture`).

Note: You can specify these parameters only with `-type latency`.

Default: `-launch`

`-logic_level`

Reports the levels (the stage count of gates) between the clock root, or generated clock root, and the capturing register.

The `-logic_level` parameter can be used only when `-type latency` is specified.

The `-logic_level` parameter cannot be specified in conjunction with the following parameters: `-launch`, `-capture`, `-from`, `-to`, `-from_clock`, `-to_clock`, `-verbose`, `-nworst`.

`-nworst` *`worst_entries`*

Specifies the number of entries to be reported per clock domain.

Default: `1`

`-source {clock_root | generated_clock}`

Reports the latency from either the master clock, or the generated clock that was derived from it.

The `-source` parameter can be used only when `-logic_level` is specified.

The `-source` parameter cannot be specified in conjunction with the following parameters: `-launch`, `-capture`, `-from`, `-to`, `-from_clock`, `-to_clock`, `-verbose`, `-nworst`.

Default: Reports the latency from the master clock (`-source clock_root`)

`-tcl_list`

Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation.

The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

`-to` *`to_list`*

Specifies the list of sequential device clock pins or ports to consider as to-pins in the generated report. If any pin in the *`to_list`* is not a sequential device clock pin, it is replaced with all the sequential device clock pins in the transitive fanout of the named pin. If there are no sequential pins in the transitive fanout of the pin, the software drops the pin from the list and generates a warning message.

`-to_clock` *`to_clock_list`*

Specifies the list of to-clocks to be used in generating the current interclock skew report. You can use this option only in an interclock skew report.

Default: Report is generated considering all clocks as to-clocks.

`-type [skew | interclock_skew | jitter | summary | latency]`

Specifies the type of report to generate.

`interclock_skew`

Reports the skew between all clocks in the design, or between clocks that you specify using the `-from_clock` and `-to_clock` parameters.

`jitter`

Reports the difference in late and early arrival times of a clock phase at a clock end point. When specified, the software generates a clock jitter report for one clock end point, for each clock in the design.

You can generate a jitter report for all clocks, or just for clocks you specify with the `-clock` parameter. You can limit the report to a specific clock end point, using the `-to` parameter.

Note: The `report_clock_timing -type jitter` parameter can be used only when the software is in On-Chip Variation (OCV) timing analysis mode. Jitter analysis does not remove any common path pessimism.

`latency`

		Reports the Network and source latency values for all clock pins in the design, or for clock pins you specify using the `-clock` parameter.
	`skew`	Reports the skew for all clocks, or for clocks that you specify using the `-clock` parameter.
	`summary`	Generates a summary report for each analyzed clock that includes the following information: ■ When setup skew is used (`-late`): ❑ maximum launch latency ❑ minimum capture latency ❑ maximum setup skew ■ When hold skew is used (`-early`): ❑ minimum launch latency ❑ maximum capture latency ❑ maximum hold skew
`-verbose`		Generates a report that contains the full clock path from the clock source to the sink pin (the clock pin of the flop or latch). You can specify the `-verbose` parameter only for skew or latency reports (`-type skew`, `-type interclock_skew`, or `-type latency`).
`-view` *`view_name`*		Generates a report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

■ The following command reports worst case skew for all clocks in a design:

```
report_clock_timing -type skew -nworst 4
**********************

Report: clock timing
        -type skew
        -setup
```

```
        -nworst 4
CRPR analysis: off
************************

Clock: Iclk

 Skew       Latency       Clock Pin

 ----------------------------------------

            532.500  r    bf5_0/flop/CK

 392.300    140.200  r    flop_out/flop/CK
```

- The following command reports interclock skew with `-from_clock`, `-to_clock`, `-from`, and `-to` parameters defined:

```
report_clock_timing -type interclock_skew -nworst 10 -from_clock CLK_L2
                    -to_clock Iclk -from clk_bf_L4/X -to clk_bf_L1/X
************************

Report: clock timing
        -type interclock_skew
        -setup
        -nworst 10
CRPR analysis: off
************************

Clocks: CLK_l2 <-> Iclk

 Skew       Latency        Clock Pin

 ----------------------------------------

                           clk_bf_L2/X

            532.500  r     bf5_0/flop/CK

                           clk

 392.300    140.200  r     flop_out/flop/CK

                           clk_bf_L2/X

            449.500  r     bf4_0/flop/CK

                           clk

 309.300    140.200  r     flop_out/flop/CK
```

- The following command reports skew for all clocks in the design including common path pessimism removal:

```
setAnalysisMode -cppr both
report_clock_timing -type skew -nworst 10
************************
```

```
Report: clock timing
        -type skew
        -setup
        -nworst 10
CRPR analysis: on
************************
Clock: Iclk

 Skew       Latency  CRPR             Clock Pin
 -------------------------------------------------------
            798.750             r     bf5_0/flop/CK
 588.450    70.100   -140.200   r     flop_out/flop/CK
            674.250             r     bf4_0/flop/CK
 463.950    70.100   -140.200   r     flop_out/flop/CK
```

- The following command generates a summary report for the clock GCLK1:

```
report_clock_timing -type summary -nworst 10 -clock GCLK1
```

```
*****************************
Report: clock timing
    -type summary
    -setup
    -nworst 10
CRPR analysis: off
*****************************
Clock: GCLK1
                         Skew   Latency         Clock Pin
-----------------------------------------------------------
Maximum Launch Latency          5.387      r    ff3/cp
Minimum Capture Latency         1.114      r    ff2/cp
Maximum Setup Skew              5.387      r    ff3/cp
                        0.20    5.187      r    ff3/cp
-----------------------------------------------------------
```

- The following command reports the Network and source latency values for the clock GCLK1:

```
report_clock_timing -type latency -nworst 10 -clock GCLK1

****************************
Report: clock timing
     -type latency
     -setup
     -nworst 10
CRPR analysis: off
****************************
Clock: GCLK1
-------------------------------------------------------------
Source         Network       Total           Clock Pin
-------------------------------------------------------------
2.000          3.187         5.187    r     ff3/cp
3.041          2.000         5.041    r     ff4/cp
-------------------------------------------------------------
```

- The following command reports the Network and source latency values for the clock GCLK1, as well as the full clock paths:

```
report_clock_timing -type latency -verbose -nworst 10 -clock GCLK1

****************************
Report: clock timing
     -type Latency
     -setup
     -launch
     -nworst 10
CRPR analysis: off
****************************
Clock: GCLK2
--------------Latency---------------------------------------
Source        Network       Total          Clock Pin
------------------------------------------------------------
4.304         1.598         5.902    r     ff6/cp

Clock Latency Path
     Source Insertion Delay          4.000
     Other End Path:
     +-------------------------------------------------+
     | Instance |     Arc     |  Cell  | Delay | Clock   |
     |          |             |        |       | Latency |
     |----------+-------------+--------+-------+---------|
     |          | CLK v       |        |       |   4.000 |
     | r1       | i v -> z v  | bufbda | 0.163 |   4.163 |
     | r2       | i v -> z v  | bufbda | 0.141 |   4.304 |
     | a1       | b v -> co v | ah01d0 | 0.271 |   4.475 |
     | r3       | i v -> z v  | bufbda | 0.174 |   4.748 |
     | r4       | i v -> z v  | bufbda | 0.142 |   4.890 |
     | r5       | i v -> z v  | bufbda | 0.141 |   4.032 |
     | r6       | i v -> z v  | bufbda | 0.155 |   5.187 |
     | r7       | i v -> z v  | bufbda | 0.139 |   5.326 |
     | r8       | i v -> zn ^ | inv0d0 | 0.364 |   5.690 |
     | adata    | a ^ -> co ^ | ah01d0 | 0.212 |   5.902 |
     | ff6      | cp ^        | sdnrq1 | 0.000 |   5.902 |
     +-------------------------------------------------+
```

- When the software is in multi-mode multi-corner timing analysis mode, the software generates a report for every active analysis view in the design:

```
report_clock_timing -type latency


****************************************
   Report: clock timing
         -type Latency
         -setup
         -launch
         -nworst 1
   CRPR analysis: off
****************************************
   Clock: PH1
   Analysis View: s2

         ---Latency---
     Source        Network          Total         Clock Pin
--------------------------------------------------------------
   Clock: PH1
   Analysis View: s1

         ---Latency---
     Source        Network          Total         Clock Pin
--------------------------------------------------------------
```

- Specify the `-view` parameter to generate a report only for a specific analysis view:

```
report_clock_timing -view s1 -type latency

****************************************
   Report: clock timing
         -type Latency
         -setup
         -launch
         -nworst 1
   CRPR analysis: off
****************************************
   Clock: PH1
   Analysis View: s1

         ---Latency---
     Source        Network          Total         Clock Pin
--------------------------------------------------------------
```

- The following command generates a textual histogram of the latency report:

```
report_clock_timing -type latency -histogram

 [Delay Chart: CLK1(clk1)]
*************************************
  Min Delay :      1.002
  Max Delay :      1.002
  Count : 1
*************************************
-----------------------------------------------------
          Delay Range
        Min        Max       Count    1
-----------------------------------------------------
         0         100          1     |o

  [Delay Chart: CLK1(genclk)]
*************************************
  Min Delay :      1.586
  Max Delay :      1.698
  Count : 3
*************************************
-----------------------------------------------------
          Delay Range
        Min        Max       Count    1
-----------------------------------------------------
         0         100          3     |ooo
```

- The following command generates the histogram report at intervals of 1:

```
 report_clock_timing -type latency -histogram -histogram_range 1

 [Delay Chart: CLK1(clk1)]
************************************
  Min Delay :      1.002
  Max Delay :      1.002
  Count : 1
************************************
-----------------------------------------------------
          Delay Range
        Min        Max       Count    1
-----------------------------------------------------
         1           2          1     |o
         0           1          0

  [Delay Chart: CLK1(genclk)]
************************************
  Min Delay :      1.586
  Max Delay :      1.698
  Count : 3
************************************
-----------------------------------------------------------
          Delay Range
        Min        Max       Count    1
-----------------------------------------------------------
         1           2          3     |ooo
         0           1          0
```

- The following command generates a latency report that includes the stage gate counts between the clock root or generated clock root, and the capturing register:

```
 report_clock_timing -type latency -logic_level

************************************************

Start point:CLK1(clk1)
-----------------------------------------------
      Delay         Count     End Point
      1.002             1     ff1/CK r

************************************************

Start point:CLK1(genclk(clk1))
-----------------------------------------------
      Delay         Count     End Point
      1.698             5     ff3/CK r
      1.586             4     ff4/CK r
      1.586             4     ff2/CK r
```

- The following command generates a latency report starting from the generated clock, and includes the stage gate counts between the generated clock and the capturing register:

```
report_clock_timing -type latency -logic_level -source generated_clock

************************************************
Start point:CLK1(clk1)
-----------------------------------------------
      Delay         Count     End Point
      1.002             1     ff1/CK r

************************************************
Start point:ff1/Q(genclk(clk1))
-----------------------------------------------
      Delay         Count     End Point
      0.385             3     ff3/CK r
      0.273             2     ff4/CK r
      0.273             2     ff2/CK r
```

- The following command generates a latency report starting from the generated clock `genclk`, and includes the stage gate counts between the generated clock and the capturing register:

```
report_clock_timing -type latency -logic_level -source generated_clock
                -clock genclk

**********************************************
Start point:ff1/Q(genclk(clk1))

-----------------------------------------------
      Delay         Count     End Point
      0.385             3     ff3/CK r
      0.273             2     ff4/CK r
      0.273             2     ff2/CK r
```

- The following command uses the `-format` parameter to customize the generated latency report:

```
report_clock_timing -type latency -verbose -format {instance arc pin cell slew
      load user_derate}

****************************************
   Report: clock timing
         -type Latency
         -setup
         -launch
         -nworst 1
   CRPR analysis: off
****************************************

   Clock: WAVE

         ---Latency---
     Source        Network          Total         Clock Pin
-------------------------------------------------------------------------
       0.000         0.352          0.352        r    BLK/BR2/CK

Clock latency Path
    Clock Rise Edge                     0.000
    = Beginpoint Arrival Time           0.000
    Other End Path:
+-------------------------------------------------------------------+
| Instance |    Arc     |  Pin  |  Cell   |  Slew |  Load |  User  |
|          |            |       |         |       |       | Derate |
|----------+------------+-------+---------+-------+-------+--------|
|          | tclk ^     | tclk  |         | 0.120 | 0.007 |        |
| CG/BC1   | A ^ -> Y ^ | Y     | BUFX2   | 0.085 | 0.011 | 1.000  |
| TC1      | A ^ -> Y ^ | Y     | BUFX2   | 0.103 | 0.015 | 1.000  |
| CG1      | B ^ -> Y ^ | Y     | AND2X4  | 0.038 | 0.004 | 1.000  |
| BLK/BC1  | A ^ -> Y ^ | Y     | BUFX2   | 0.081 | 0.010 | 1.000  |
| BLK/BR2  | CK ^       | CK    | DFFHQX1 | 0.081 | 0.010 | 1.000  |
+-------------------------------------------------------------------+
```

- The following command reports the worst case jitter for all clocks in the design:

```
report_clock_timing -type jitter -nworst 2
****************************************
  Report: clock timing
        -type jitter
        -setup
```

```
       -nworst 2
  CRPR analysis: off
****************************************

  Clock: CLK1

      Jitter      Latency(Late)    Latency(Early)            Clock Pin
--------------------------------------------------------------------------
       0.186         0.350      r            0.164      r    i_17/CK
       0.118         0.227      r            0.109      r    i_15/CK

  Clock: CLK0

      Jitter      Latency(Late)    Latency(Early)            Clock Pin
--------------------------------------------------------------------------
       0.387         0.508      r            0.120      r    i_5/CK
       0.178         0.336      r            0.159      r    i_7/CK
```

- The following command reports the worst case jitter for the clock CLK1:

```
report_clock_timing -type jitter -clock CLK1 -nworst 2
****************************************
  Report: clock timing
        -type jitter
        -setup
        -nworst 2
  CRPR analysis: off
****************************************

  Clock: CLK1

      Jitter      Latency(Late)    Latency(Early)            Clock Pin
--------------------------------------------------------------------------
       0.186         0.350      r            0.164      r    i_17/CK
       0.118         0.227      r            0.109      r    i_15/CK
```

- The following command reports the worst case jitter for the specified clock end point:

```
report_clock_timing -type jitter -to i_15/CK
****************************************
  Report: clock timing
        -type jitter
        -setup
        -nworst 2
```

```
  CRPR analysis: off
****************************************

  Clock: CLK1

      Jitter      Latency(Late)    Latency(Early)            Clock Pin
-------------------------------------------------------------------------
       0.118         0.227      r            0.109      r    i_15/CK
```

report_clocks

```
report_clocks
     [-description]
     [-arrival_points]
     [-phase_shift_table]
     [-total_shift_table]
     [-uncertainty_table]
     [-adjustment_table]
     [-delay_adjustment_table]
     [-source_insertion]
     [-insertion]
     [-generated]
     [-clocks clk_signame | clk_signame_list]
     [-view view_name]
     [{> | >>} filename | -tcl_list]
```

Reports information on clocks created with the `create_clock` command. The report includes information about all clock waveforms, clock arrival points, and clock uncertainties.

Use the `-description`, `-arrival_points`, `-uncertainty_table`, and the `*_table` parameters to specify different parts of the report. If one of these options is supplied, only the section that the option corresponds to is reported. If other options are supplied, those sections are also reported.

Use the `-generated` parameter to report information on generated clocks in the design.

Tip

The `report_clocks` command only reports constraints with respect to clock waveforms. Use the `report_ports -type` command for constraints applied to clock pins.

Parameters

`{> | >>}` *`filename`*

When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it.

When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the `.gz` extension to the file name.

Default: Report is displayed on standard output without being saved.

`-adjustment_table`

Reports two tables that detail the hold and setup cycle adjustments between clock waveforms as set by the `set_multicycle_path` command.

Note: If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.

`-arrival_points`

Reports only the clock arrival points. A clock arrival point is the pin where the `create_clock` command has been asserted. If you specify the `-arrival_points` parameter with the `-generated` parameter, the software reports the clock arrival points of the generated clocks.

`-clocks` *`clk_signame`*|*`clk_signame_list`*

Requests information on a specific ideal (or generated) clock. If a list is given, only information about those clocks is reported. You can use wildcards in the names.
Default: All clocks (except generated clocks) are reported. Use the `-generated` option to also report generated clocks.

`-delay_adjustment_table`

	Reports two tables that detail the early and late path delay adjustments between clock waveforms as set by the `set_max_delay` and `set_min_delay` commands. **Note:** If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.
`-description`	Reports only the description of the specified clock waveform(s).
`-generated`	Includes information on generated clocks in the report, such as source and master clock of the generated clock, and other information specified in the `create_generated_clock` constraints. The report uses the internal clock names created by the system, if you did not specify new clock names while creating these clocks.
`-insertion`	Reports the network insertion delays specified on clock waveforms. The values in the table are in `min:typ:max` format.
`-phase_shift_table`	Reports two tables that detail the phase shift between clock waveforms. One table shows the phase shifts in late mode and the other in early mode. **Note:** If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.
`-source_insertion`	Reports the source insertion delays specified on clock waveforms.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together. See Example 32-16 on page 1728 for an example of the `-tcl_list` output.
`-total_shift_table`	

Reports two tables (late mode and early mode) that detail the total relationship between clock waveforms.

Note: If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.

`-uncertainty_table`

Reports two tables (late mode and early mode) that detail the uncertainty between clock waveforms as specified by the `set_clock_uncertainty` command.

Note: If specific ideal (or generated) clocks are requested with the `-clocks` *`clock_name_list`* option, then only the relationships between those clock signals are reported.

`-view` *`view_name`* Generates a clock report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- The following command displays the default report:

```
report_clocks
```

Example 32-8 report_clocks Report

```
|                        Clock Descriptions                          |
|----------------------------------------------------------------------|
|       |        |        |       |       |       Attributes       |
|-------+--------+--------+-------+-------+------------------------|
| Clock | Source | Period |  Lead | Trail | Generated | Propagated |
|  Name |        |        |       |       |           |            |
|-------+--------+--------+-------+-------+-----------+------------|
|  clk  |  clk   |  8.500 | 0.000 | 4.250 |     n     |     n      |
+----------------------------------------------------------------------+
+-------------------------------------------------+
|    Total Clock To Clock Relationship (Late)     |
|-------------------------------------------------|
|  From |  To   |  L->L |  L->T  |  T->L |  T->T  |
|-------+-------+-------+--------+-------+--------|
|   clk |   clk | 8.100 | -0.400 | 8.100 | 8.100 |
+-------------------------------------------------+
+-------------------------------------------------+
|   Total Clock To Clock Relationship (Early)     |
|-------------------------------------------------|
|  From |  To   |  L->L |  L->T  |  T->L |  T->T  |
|-------+-------+-------+--------+-------+--------|
|   clk |   clk | 0.400 | -8.100 | 0.400 | 0.400 |
+-------------------------------------------------+
```

- The following command displays only the uncertainty information:

  ```
  report_clocks -uncertainty_table
  ```

Example 32-9 report_clocks -uncertainty_table Report

```
+-----------------------------------------------+
|       Clock To Clock Uncertainty (Late)       |
|-----------------------------------------------|
|  From |  To   |  L->L |  L->T |  T->L |  T->T |
|-------+-------+-------+-------+-------+-------|
| iclk1 | iclk1 |  0.00 |  0.00 |  0.00 |  0.00 |
| iclk1 | iclk2 |  1.70 |  1.70 |  1.70 |  1.70 |
| iclk2 | iclk1 |  0.00 |  0.00 |  0.00 |  0.00 |
| iclk2 | iclk2 |  0.00 |  0.00 |  0.00 |  0.00 |
+-----------------------------------------------+

+-----------------------------------------------+
|      Clock To Clock Uncertainty (Early)       |
|-----------------------------------------------|
|  From |  To   |  L->L |  L->T |  T->L |  T->T |
|-------+-------+-------+-------+-------+-------|
| iclk1 | iclk1 |  0.00 |  0.00 |  0.00 |  0.00 |
| iclk1 | iclk2 | -2.70 | -2.70 | -2.70 | -2.70 |
| iclk2 | iclk1 |  0.00 |  0.00 |  0.00 |  0.00 |
| iclk2 | iclk2 |  0.00 |  0.00 |  0.00 |  0.00 |
+-----------------------------------------------+
```

- Consider the following clock definition:

  ```
  create_clock -name "clock1" -add -period 6 -waveform {0 3}
  ```

  When you specify the `report_clocks` command, the following phase shift table is displayed:

```
+-----------------------------------------------+
|       Total Clock To Clock Relationship (Late)|
|-----------------------------------------------|
|  From |  To   |  L->L |  L->T |  T->L |  T->T |
|-------+-------+-------+-------+-------+-------|
| clock1| clock1| 6.000 |  0.000| 6.000| 6.000|
| +---------------------------------------+
```

In this case, `L->L` has phase shift 6 which means that the path starts from the leading edge of the first period and ends at the leading edge of the second period.

`L->T` has phase shift `0` which means that the path starts from the leading edge of the first period and ends at the trailing edge of the first period.

`T->L` has phase shift 6 which means that the path starts from the trailing edge of first period and ends at the leading edge of the second period.

`T->T` has phase shift 6 which means that the path starts from the trailing edge of the first period and ends at the trailing edge of the second period.

You can use the following command to add uncertainty:

```
set_clock_uncertainty 0.2 clock1
```

The phase shift and total shift table values will be different. The following command displays the changed shift values as given below:

```
report_clocks -phase_shift_table -total_shift_table
```

```
+-----------------------------------------------+
|       Total Clock To Clock Relationship (Late)|
|-----------------------------------------------|
|  From |  To   |  L->L |  L->T |  T->L |  T->T |
|-------+-------+-------+-------+-------+-------|
| clock1| clock1| 5.800 | -0.200| 5.800| 5.800|
| +---------------------------------------+
```

You can see that the uncertainty is substracted from the phase shift.

Consider the following example path:

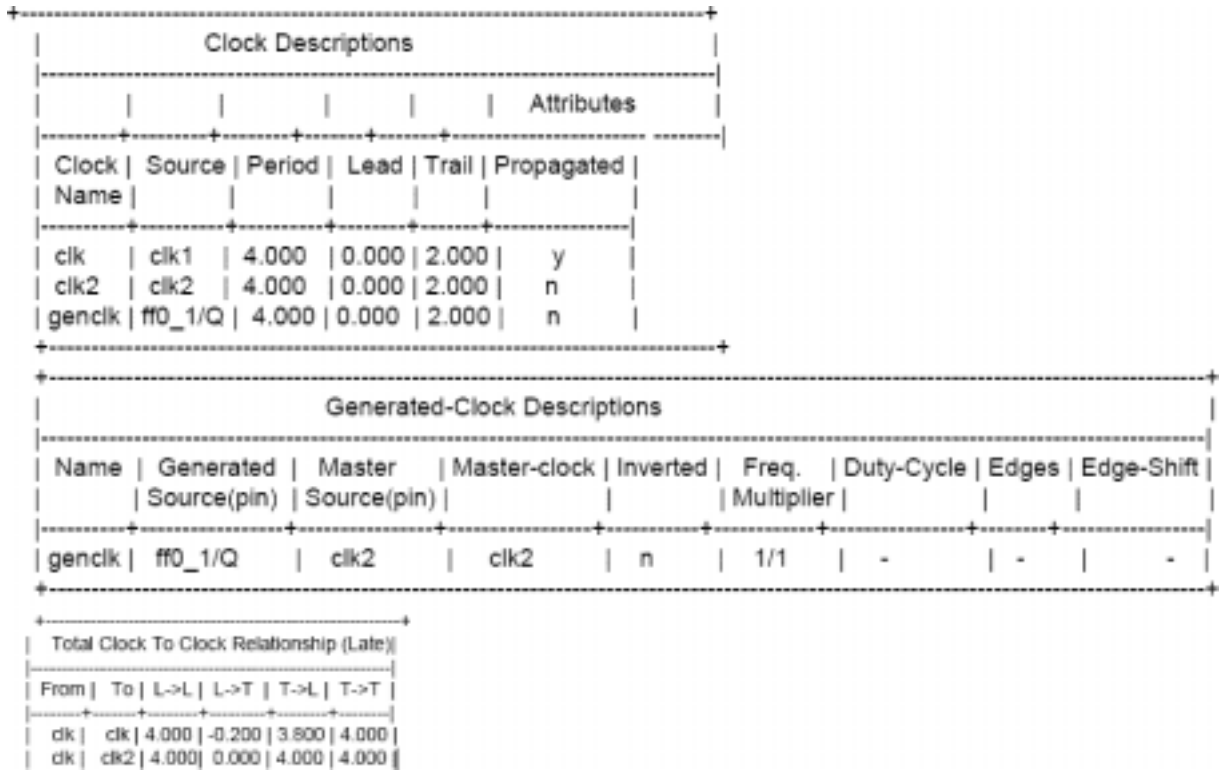
```
Path 1: MET Late External Delay Assertion
Endpoint:   odata (v) checked with  leading edge of 'std_clk'
Beginpoint: idata (v) triggered by trailing edge of 'div_2_clk'
Other End Arrival Time   0.000
- External Delay         0.000
+ Phase Shift           12.000
- Uncertainty            0.100
= Required Time         11.900
- Arrival Time           7.500
= Slack Time             4.400
     Clock Rise Edge            6.000
     + Input Delay              0.000
     = Beginpoint Arrival Time  6.000

+-------------------------------------------------------------------------+
| Instance| Arc           |  Cell   | Delay | Arrival Time | Required Time|
|-------------------------------------------------------------------------|
|         |idata v        |         |       |  6.000       | 10.400       |
| buf1    |H01 v -> N01 v | TDBUFX2 | 1.500 |  7.500       | 11.900       |
|         |odata v        |         | 0.000 |  7.500       | 11.900       |
|+-----------------------------------------------------------------------+
```

Here the begin point `div_2_clk` to end point `std_clk` shows a phase shift value of 12.00. This means that if a path starts from the falling edge of `div_2_clk`, and ends at the rising edge of `std_clk`, the phase shift is 12.00. In other words, when the timing (setup time) is reported, the required time at the end point begins with 12.00 (in addition to adjustments like setup time, time borrowing, uncertainty, etc). However, the arrival time of this path starts at 6.00.

- The following command displays a report from a design with a generated clock:

```
report_clocks -generated
```

Example 32-10 report_clocks -generated Report



Clock Descriptions

Clock Name	Source	Period	Lead	Trail	Attributes: Propagated
clk	clk1	4.000	0.000	2.000	y
clk2	clk2	4.000	0.000	2.000	n
genclk	ff0_1/Q	4.000	0.000	2.000	n

Generated-Clock Descriptions

Name	Generated Source(pin)	Master Source(pin)	Master-clock	Inverted	Freq. Multiplier	Duty-Cycle	Edges	Edge-Shift
genclk	ff0_1/Q	clk2	clk2	n	1/1	-	-	-

Total Clock To Clock Relationship (Late)

From	To	L->L	L->T	T->L	T->T
clk	clk	4.000	-0.200	3.800	4.000
clk	clk2	4.000	0.000	4.000	4.000

- When the software is in multi-mode multi-corner timing analysis mode, the software reports the clock information for each active analysis view:

```
report_clocks > rptClocks.txt
```

```
+-------------------------------------------------------------------------------+
|                              Clock Descriptions                               |
|-------------------------------------------------------------------------------|
|       |        |              |        |       |       |       Attributes |
|-------+--------+--------------+--------+-------+-------+------------------|
| Clock | Source |    View      | Period |  Lead | Trail | Generated | Propagated|
|  Name |        |              |        |       |       |           |           |
|-------+--------+--------------+--------+-------+-------+-----------+------|
|  PH1  |  clk1  |   clk1-typ   | 10.000 | 0.000 | 5.000 |     n     |   y  |
|  PH1  |  clk1  | clk1-reg2reg | 10.000 | 0.000 | 5.000 |     n     |   y  |
|  PH1  |  clk1  |  clk1-slow   | 10.000 | 0.000 | 5.000 |     n     |   y  |
|  PH1  |  clk1  |  clk1-fast   | 10.000 | 0.000 | 5.000 |     n     |   y  |
```

```
|  PH3  |        |  clk1-fast   | 10.000 | 0.000 | 5.000 |     n     |   n  |
|  PH3  |        |  clk1-slow   | 10.000 | 0.000 | 5.000 |     n     |   n  |
|  PH3  |        |   clk1-typ   | 10.000 | 0.000 | 5.000 |     n     |   n  |
+-------------------------------------------------------------------------------+
+-----------------------------------------------------------------------+
|               Total Clock To Clock Relationship (Late)                |
|-----------------------------------------------------------------------|
|  From |  To   |     View     |  L->L   |  L->T   |  T->L   |  T->T   |
|-------+-------+--------------+---------+---------+---------+---------|
|   PH1 |   PH1 | clk1-reg2reg |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH1 |   PH1 |    clk1-slow | -10.000 | -20.000 | -10.000 | -10.000 |
|   PH1 |   PH1 |    clk1-fast | -10.000 | -20.000 | -10.000 | -10.000 |
|   PH1 |   PH1 |     clk1-typ | -10.000 | -20.000 | -10.000 | -10.000 |
|   PH1 |   PH3 |    clk1-fast |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH1 |   PH3 |     clk1-typ |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH1 |   PH3 |    clk1-slow |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH3 |   PH1 |    clk1-fast |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH3 |   PH1 |    clk1-slow |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH3 |   PH1 |     clk1-typ |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH3 |   PH3 |    clk1-fast |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH3 |   PH3 |     clk1-typ |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH3 |   PH3 |    clk1-slow |  10.000 |   0.000 |  10.000 |  10.000 |
+-----------------------------------------------------------------------+
+-----------------------------------------------------------------------+
|              Total Clock To Clock Relationship (Early)                |
|-----------------------------------------------------------------------|
|  From |  To   |     View     |  L->L  |  L->T   |  T->L  |  T->T  |
|-------+-------+--------------+--------+---------+--------+--------|
|   PH1 |   PH1 | clk1-reg2reg |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH1 |   PH1 |    clk1-slow | 20.000 |  10.000 | 20.000 | 20.000 |
|   PH1 |   PH1 |    clk1-fast | 20.000 |  10.000 | 20.000 | 20.000 |
|   PH1 |   PH1 |     clk1-typ | 20.000 |  10.000 | 20.000 | 20.000 |
|   PH1 |   PH3 |    clk1-fast |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH1 |   PH3 |     clk1-typ |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH1 |   PH3 |    clk1-slow |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH3 |   PH1 |    clk1-fast |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH3 |   PH1 |    clk1-slow |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH3 |   PH1 |     clk1-typ |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH3 |   PH3 |    clk1-fast |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH3 |   PH3 |     clk1-typ |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH3 |   PH3 |    clk1-slow |  0.000 | -10.000 |  0.000 |  0.000 |
```

```
+-------------------------------------------------------------------+
```

- Specify the `-view` parameter to generate clock information for a specific analysis view only:

```
report_clocks -view clk1-slow >> rptClocks.txts
+-------------------------------------------------------------------------------+
|                              Clock Descriptions                               |
|-------------------------------------------------------------------------------|
|       |        |           |        |       |       |       Attributes      |
|-------+--------+-----------+--------+-------+-------+-----------------------|
| Clock | Source |   View    | Period |  Lead | Trail | Generated | Propagated|
|  Name |        |           |        |       |       |           |           |
|-------+--------+-----------+--------+-------+-------+-----------+-----------|
|  PH1  |  clk1  | clk1-slow | 10.000 | 0.000 | 5.000 |     n     |     y     |
|  PH3  |        | clk1-slow | 10.000 | 0.000 | 5.000 |     n     |     n     |
+-------------------------------------------------------------------------------+
+-------------------------------------------------------------------+
|             Total Clock To Clock Relationship (Late)              |
|-------------------------------------------------------------------|
|  From |  To   |   View    |  L->L   |  L->T   |  T->L   |  T->T   |
|-------+-------+-----------+---------+---------+---------+---------|
|   PH1 |   PH1 | clk1-slow | -10.000 | -20.000 | -10.000 | -10.000 |
|   PH1 |   PH3 | clk1-slow |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH3 |   PH1 | clk1-slow |  10.000 |   0.000 |  10.000 |  10.000 |
|   PH3 |   PH3 | clk1-slow |  10.000 |   0.000 |  10.000 |  10.000 |
+-------------------------------------------------------------------+
+-----------------------------------------------------------+
|           Total Clock To Clock Relationship (Early)       |
|-----------------------------------------------------------|
|  From |  To   |   View    |  L->L  |  L->T   |  T->L  |  T->T  |
|-------+-------+-----------+--------+---------+--------+--------|
|   PH1 |   PH1 | clk1-slow | 20.000 |  10.000 | 20.000 | 20.000 |
|   PH1 |   PH3 | clk1-slow |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH3 |   PH1 | clk1-slow |  0.000 | -10.000 |  0.000 |  0.000 |
|   PH3 |   PH3 | clk1-slow |  0.000 | -10.000 |  0.000 |  0.000 |
+-----------------------------------------------------------+
```

Related Information

`create clock`

create_generated_clock

set_clock_uncertainty

report_constraint

```
report_constraint
     [-all_violators]
     [-verbose]
     [-locv]
     [-early | -late]
     [-path_type slack_only]
     [-max_delay]
     [-min_delay]
     [-max_capacitance]
     [-max_transition]
     [-max_fanout]
     [-min_pulse_width]
     [-min_period]
     [-recovery]
     [-removal]
     [-clock_gating_setup]
     [-clock_gating_hold]
     [-significant_digits digits]
     [-group]
     [-connection_class]
     [-view viewName]
     [{> | >>} filename]
```

Reports constraint information of current design. The constraints information includes information such as whether or not constraint is violated, by how much amount, and the worst violator object.

The following timing checks are reported when you use the `report_constraint` command:

- Setup and hold, including path exception
- DRV
- Clock and asynchronous checks including min_pulse_width, min_period, recovery, removal, clock_gating_setup, clock_gating_hold.

Note: If you do not specify either of the `-early` or `-late` parameters, the command reports all of the constraints by switching between setup and hold modes, which can incur some runtime penalty. Use the `-early` and `-late` parameters to restrict reporting to the constraints that are consistent with the current `setAnalysisMode` setting to avoid the setup and hold switching delay.

Parameters

`{> | >>}` *`filename`*

When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it.

When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the `.gz` extension to the file name.

Default: Report is displayed on standard output without being saved.

`-all_violators`

Reports negative slack values for timing checks. The supported check types are:

- setup
- hold
- clock_gating_setup
- clock_gating_hold
- recovery
- removal
- clock_period
- pulse_width
- pins that exceed maximum capacitance and transition constraints
- pins that exceed the maximum fanout constraints

You can use the `-all_violators` parameter with parameters for reporting specific check types to generate report for that particular check type only. For example, you can use `-all_violators` with `-max_delay` to report setup checks and path delays from set_max_delay.

`-clock_gating_hold`	Reports only the paths that end at the clock_gating_hold timing check.
`-clock_gating_setup`	Reports only the paths that end at the clock_gating_setup timing check.
`-connection_class`	Reports violations if the driving pin and receiving pin of a net have different connection_class attributes defined.
`-early`	Limits the reporting to the following constraints: min_delay, removal, and clock_gating_hold. Specify `-early` to limit reporting to the constraints analyzed under `setAnalysisMode -checkType hold.` *Default:* Reports all of the timing checks in the following order, based on whether the software is in setup or hold analysis mode: ■ Setup analysis mode: Reports checks specified with the `-late` parameter first, then checks specified with the `-early` parameter. ■ Hold analysis mode: Reports checks specified with the `-early` parameter first, then checks specified with the `-late` parameter.
`-group`	Groups the endpoints in the generated report by clock names. This parameter only works for endpoints reported when you use the `-max_delay`, `-min_delay`, or `-all_violators` parameters. Each endpoint is reported once per group. The `report_constraint` command uses path groups created by the `timing_path_groups_for_clocks` global variable. If this variable is not set, then the `report_constraint` command creates path groups.

`-late`	Limits the reporting to the following constraints: clock_gating_setup, max_capacitance, max_delay, max_fanout, max_transition, min_period, min_pulse_width, and recovery. Specify `-late` to limit reporting to the constraints analyzed under `setAnalysisMode-checkType setup`. *Default:* Reports all of the timing checks in the following order, based on whether the software is in setup or hold analysis mode: ■ Setup analysis mode: Reports checks specified with the `-late` parameter first, then checks specified with the `-early` parameter. ■ Hold analysis mode: Reports checks specified with the `-early` parameter first, then checks specified with the `-late` parameter.
`-locv`	Generates a report that displays the summary of the LOCV slack calculation, considering LOCV derating at various endpoints. If you specify the `-verbose` option with `-locv`, the resultant report displays the full data path with accompanying required time and slack calculation. The report thus obtained is equivalent to `report_timing -retime locv`. You can customize the format of this report by using the `set table style` command.
`-max_capacitance`	Reports all pins that exceed the maximum capacitance constraints in the timing library and timing constraints file.
`-max_delay`	Reports all of the setup checks, including regular, clock_gating_cetup, and data-to-data setup checks. Also reports path delays from set_max_delay.
`-max_fanout`	Reports all the pins that exceed the maximum fanout constraints in the timing library and timing constraints file.
`-max_transition`	Reports all pins that exceed the transition constraints in the timing constraints file.
`-min_delay`	Reports all of the hold checks, including regular, clock_gating_hold, and data-to-data hold checks. Also reports path delays from set_min_delay.

`-min_period`	Reports only the paths that end at the clock_period timing check.
`-min_pulse_width`	Reports only the paths that end at the pulse_width timing check.
`-path_type slack_only`	Specifies the format of the report by path type, slack_only. The resulting report contains only endpoint or name of the net and slack value.
`-recovery`	Reports only the paths that end at the recovery timing check.
`-removal`	Reports only the paths that end at the removal timing check.
`-significant_digits` *`digits`*	Specifies the number of digits appearing after the decimal point in the generated report. This option sets the `report_precision` global variable to the specified value.
`-verbose`	Generates a report that displays the full data path with accompanying required time and slack calculation. This report is equivalent to the default `report_timing` report. If you specify `-verbose`, you can use the `set_table_style` command to customize the format of your report.
`-view` *`viewName`*	Generates a report containing endpoint and other constraint information for the specified analysis view. You can only specify this parameter when the software is in multi-mode multi-corner analysis mode. *Default:* Generates a report containing the worst endpoint and constraint information based on all of the analysis views. **Note:** You cannot use the `-view` parameter to report transition, capacitance, and fanout violation information per view. These violations are reported independently of analysis views.

Examples

- The following command generates report formatted by path type, `slack_only`, and directs the output in the report file, `summarySlackOnly.rpt`:

```
   report_constraint -path_type slack_only > summarySlackOnly.rpt

      Check type  Endpoint                                          Slack
      ----------  --------                                          -------
 max_delay/setup  TDSP_DS_CS_INST/i_8/B f                           1.406  MET
  min_delay/hold  TEST_CONTROL_INST/i_0218/B f                      -4.811  VIOLATED
        recovery  SPI_INST/bit_cnt_reg_3/RN r                       2.755  MET
         removal  -                                                 0.000  MET
      min_period  -                                                 0.000  MET
min_pulse_width  RAM_256x16_TEST_INST_RAM_256x16_INST/wr r          0.016  MET

      Check type  Net Name                                          Slack
      ----------  --------                                          -----
 max_transition  RESULTS_CONV_INST/n_209                            -14.645r VIOLATED
max_capacitance  -                                                  0.000  MET
      max_fanout  -                                                 0.000  MET
```

- The following command generates report formatted by path type, `slack_only`, and groups the endpoints in the generated report by clock names:

```
report_constraint -path_type slack_only
                    -all_violators -group > allViolSlackOnly.rpt

Check type : max_delay/setup ('clock_gating_default' group)
Endpoint                                                Slack
--------                                                -----
-                                                       0.000  MET


Check type : max_delay/setup ('vclk1' group)
Endpoint                                                Slack
--------                                                -----
TDSP_CORE_INST/EXECUTE_INST/acc_reg_18/D f              -23.555  VIOLATED
RESULTS_CONV_INST/gt_reg/D r                            -0.236  VIOLATED
TDSP_CORE_INST/EXECUTE_INST/p_reg_1/D r                 -0.128  VIOLATED
TDSP_CORE_INST/EXECUTE_INST/ar1_reg_2/D f               -0.103  VIOLATED
TDSP_CORE_INST/EXECUTE_INST/ar1_reg_3/D f               -0.100  VIOLATED
TDSP_CORE_INST/EXECUTE_INST/ar1_reg_5/D f                -0.095  VIOLATED
TDSP_CORE_INST/EXECUTE_INST/ar1_reg_6/D f               -0.092  VIOLATED
TDSP_CORE_INST/EXECUTE_INST/ar1_reg_0/D f               -0.089  VIOLATED
TDSP_CORE_INST/EXECUTE_INST/ar1_reg_1/D f               -0.088  VIOLATED
```

```
TDSP_CORE_INST/EXECUTE_INST/ar1_reg_4/D f              -0.086  VIOLATED
TDSP_CORE_INST/EXECUTE_INST/ar1_reg_7/D f              -0.084  VIOLATED

Check type : max_delay/setup ('vclk2' group)
Endpoint                                               Slack
--------                                               -----
-                                                      0.000  MET

Check type : min_delay/hold ('clock_gating_default'
group)
Endpoint                                               Slack
--------                                               -----
TEST_CONTROL_INST/i_0218/B f                           -4.811  VIOLATED
TEST_CONTROL_INST/i_4/B r                              -4.295  VIOLATED
TEST_CONTROL_INST/i_3/B r                              -4.095  VIOLATED

Check type : min_delay/hold ('vclk1' group)
Endpoint                                               Slack
--------                                               -----
-                                                      0.000  MET

Check type : min_delay/hold ('vclk2' group)
Endpoint                                               Slack
--------                                               -----
-                                                      0.000  MET

Check type : recovery
Endpoint                                               Slack
--------                                               -----
-                                                      0.000  MET

Check type : removal
Endpoint                                               Slack
--------                                               -----
-                                                      0.000  MET

Check type : min_period
Endpoint                                               Slack
```

```
--------                                                  -----
-                                                         0.000  MET

Check type : min_pulse_width
Endpoint                                                  Slack
--------                                                  -----
-                                                         0.000  MET

Check type : max_transition
Net Name                                                  Slack
--------                                                  -----
RESULTS_CONV_INST/n_209                                   -14.645r VIOLATED
t_addrs[1]                                                -0.013r VIOLATED
TDSP_CORE_INST/ALU_32_INST/nbus_1119[19]                  -0.008f VIOLATED
upcm[7]                                                   -0.007r VIOLATED
TDSP_CORE_INST/ALU_32_INST/nbus_1119[20]                  -0.003f VIOLATED
TDSP_CORE_INST/ALU_32_INST/i_51412_n_2239342              -0.003f VIOLATED

Check type : max_capacitance
Net Name                                                  Slack
--------                                                  -----
-                                                         0.000  MET

Check type : max_fanout
Net Name                                                  Slack
--------                                                  -----
-                                                         0.000  MET
```

- The following command generates a summary report:

```
report_constraint > summary.rpt
```

```
     Check type Begin Point End Point Slack   Status/Cause
     ---------- ----------- --------- ------  -------
max_delay/setup      bit1 r    g1/D r -0.2500 VIOLATED Setup Check with Pin g1/CP
 min_delay/hold      g3/Q f   g13/D f 0.2436  MET Hold Check with Pin g13/CP
     min_period      -          -     0.0     MET
min_pulse_width      clk1 f    clk1 f -0.5000 VIOLATED User PulseWidth Check with
                                              Pin clk1
```

```
     Check type    Net Name Slack     Status
     ---------- ----------- -----     ------
 max_transition      -       0.0      MET
max_capacitance      -       0.0      MET
     max_fanout      -       0.0      MET
```

- The following command generates negative slack report for timing checks:

```
report_constraint -all_violators > allViol.rpt
```

```
Check type : max_delay/setup
    Begin Point End Point  Slack    Status/Cause
     ---------- ---------  -------- -------
           ci f     g3/D f  -0.2500 VIOLATED Setup Check with Pin g3/CP
           ci r     g3/D f  -0.2500 VIOLATED Setup Check with Pin g3/CP
         bit2 f     g2/D f  -0.2500 VIOLATED Setup Check with Pin g2/CP
         bit2 r     g2/D f -0.2500  VIOLATED Setup Check with Pin g2/CP
         bit1 f     g1/D f -0.2500  VIOLATED Setup Check with Pin g1/CP
         bit1 r     g1/D f -0.2500  VIOLATED Setup Check with Pin g1/CP

Check type : min_delay/hold
    Begin Point End Point  Slack    Status/Cause
     ---------- ---------  -------- -------
               -     -     0.0      MET

Check type : min_period
    Begin Point End Point  Slack    Status/Cause
     ---------- ---------  -------- -------
               -     -     0.0      MET

Check type : min_pulse_width
    Begin Point End Point  Slack    Status/Cause
     ---------- ---------  -------- -------
         clk1 f   clk1 f  -0.5000  VIOLATED User PulseWidth Check with Pin clk1
         clk1 r   clk1 r  -0.5000  VIOLATED User PulseWidth Check with Pin clk1
         clk1 f  g13/CP f -0.5000  VIOLATED User PulseWidth Check with Pin g13/CP
         clk1 r  g13/CP r -0.5000  VIOLATED User PulseWidth Check with Pin g13/CP
         clk1 f  g12/CP f -0.5000  VIOLATED User PulseWidth Check with Pin g12/CP
```

```
        clk1 r  g12/CP r  -0.5000  VIOLATED User PulseWidth Check with Pin g12/CP
        clk1 f   g3/CP f  -0.5000  VIOLATED User PulseWidth Check with Pin g3/CP
        clk1 r   g3/CP r  -0.5000  VIOLATED User PulseWidth Check with Pin g3/CP
        clk1 f   g2/CP f  -0.5000  VIOLATED User PulseWidth Check with Pin g2/CP
        clk1 r   g2/CP r  -0.5000  VIOLATED User PulseWidth Check with Pin g2/CP
        clk1 f   g1/CP f  -0.5000  VIOLATED User PulseWidth Check with Pin g1/CP
        clk1 r   g1/CP r  -0.5000  VIOLATED User PulseWidth Check with Pin g1/CP

    Check Type Net Name   Slack    Status
    ---------- --------- -------- -------
max_transition     -     0.0      MET
max_capacitanc     -     0.0      MET
            e
    max_fanout     -     0.0      MET
```

report_cppr

```
report_cppr
     -from pin_or_port
     -to pin_or_port
     [-from_clock clkname]
     [-to_clock clkname]
     [-early | -late]
     [-view view_name]
     [> filename]
```

Reports the clock reconvergence pessimism value at the common branching point of the early and late paths in the clock network of the specified data path.

The `report_cppr` command reports separate values for both opening and closing edges for paths with latches as destinations. If both source and destination flops are triggered by multiple clocks, then one report is generated for each separate clock.

When the software is in multi-mode multi-corner analysis mode, `report_cppr` reports the results for each active analysis view.

To use the `report_cppr` command, you must first set the following globals:

- `timing_cppr_transition_sense`
- `timing_cppr_threshold_ps`

Parameters

`> filename`	Writes the report to the specified file name. If the file already exists, the software overwrites it. The *`filename`* parameter must be the last argument in the list. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.

`-early`	Reports the hold analysis clock reconvergence pessimism value. You can specify this parameter when the software is in hold analysis mode, or in simultaneous setup and hold analysis mode (`set_global timing_enable_simultaneous_setup_hold_mode true`).
`-from` *`pin_or_port`*	Reports the clock reconvergence pessimism value for paths starting from the specified clock pin of the source flop, or from the specified port.
`-from_clock` *`clkname`*	Reports the clock reconvergence pessimism value for paths with the specified triggering clock. Use this parameter when there is more than one clock triggering the source flop.
`-late`	Reports the setup analysis clock reconvergence pessimism value. You can specify this parameter when the software is in setup analysis mode, or in simultaneous setup and hold analysis mode (`set_global timing_enable_simultaneous_setup_hold_mode true`).
`-to` *`pin_or_port`*	Reports the clock reconvergence pessimism value for paths ending at the specified clock pin of the destination flop, or at the specified port.
`-to_clock` *`clkname`*	Reports the clock reconvergence pessimism value for paths with the specified capturing clock. Use this parameter when there is more than one clock capturing the destination flop.
`-view` *`view_name`*	Reports the clock reconvergence pessimism values for paths in the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- When the software is in multi-mode multi-corner timing analysis mode, you can specify the `-view` parameter to report the CRP values for a specific analysis view:

```
report_cppr -from RS/CK -to RT/CK -view s1

 Start Clock Pin: RS/CK (Triggered by rising  edge of clock PH1)
 End Clock Pin: RT/CK (Triggered by rising  edge of clock PH1)
 Analysis Type: Setup
 Analysis View: s1
 Common Branch Point : C2/Y Launching
 Edge Type At Common Point : Rising Capturing
 Edge Type At Common Point : Rising
 Pessimism Threshold Value :    0.020
       +--------------------------------------+
       | Arrival |  Late  | Early | Pessimism |
       |  Times  |        |       |           |
       |---------+--------+-------+-----------|
       |  Rise   | 10.000 | 6.000 |   4.000   |
       |  Fall   | 10.000 | 6.000 |   4.000   |
       +--------------------------------------+
 Edge Selection Mode : Normal
 Edge Selection : Match, Using Rise
 CPPR Adjustment :   4.000
```

report_design

```
report_design
     [{> | >>} filename | -tcl_list]
```

Reports the operating conditions and active design rules for the current design.

Parameters

`{> | >>} filename`	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together. See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

Example

- The following is a sample report generated using the `report_design` command:

```
encounter 16> report_design
```

Design Property	Value
Design Name	dsp_core
Operating Condition Name	slow
Process	1.000
Voltage	1.620
Temperature	125.000
Tree Type	balanced

Design Rule	Value
Max Transition	NA
Min Transition	NA
Max Capacitance	NA
Min Capacitance	NA
Max Fanout	24.000

report_inactive_arcs

```
report_inactive_arcs
     [instance_or_port_list]
     [-delay_arcs_only | -check_arcs_only]
     [-type disable_type]
     [-include_net_arcs]
     [-view viewName]
     [{> | >>} filename]
```

Reports information about disabled timing and timing check arcs in the design. Reports all arcs disabled due to user-specified exceptions such as `set_disable_timing` or `set_case_analysis`, as well as information about arcs disabled by constant propagation during timing analysis, snipped loop arcs, and arcs disabled in the timing library.

Parameters

`{> | >>} filename`	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-check_arcs_only`	Lists only disabled check arcs. The `check_arcs` are disabled by using the `set_disable_timing` command. *Default*: Both disabled timing delay arcs and check arcs are listed.
`-delay_arcs_only`	Lists only disabled timing delay arcs.
`-include_net_arcs`	Includes disabled net arcs in the report.
instance_or_port_list	

Reports disabled timing arcs for the specified list of instances or ports. If you specify a list of instances, the software reports all disabled arcs of the specified cells. If you specify a list of ports, the software reports all disabled arcs directly connected to the ports.

Default: Reports all disabled net and cell arcs.

`-type` *`disable_type`* Lists only the disabled arcs that belong to this category of disabled arcs. The `disable_type` option restricts the report to only those arcs which are disabled due to a particular timing analysis step, such as constant propagation, specified disables, or loop arc disables. The following are valid *`disable_type`* values:

- `check_types`

  Setup, Hold, Recovery, Removal, PulseWidth, ClockGatingSetup, ClockGatingHold, ClockGatingPulseWidth, ClockPeriod, Skew, ClockSeparation, NoChange. These types apply to the disabled check arcs and are listed under the *CheckType* field if the disabled arc is a timing check arc. For delay arcs the *CheckType* field is empty.

- `const`

  Propagated constant — arcs disabled due to a constant value. The constant value can apply to one of the arc nets/pins, to side inputs which disable a condition for the arc, to nets that disable an active mode for the arc, or disable a default conditional arc.

- `disable`

  Defined by using the `set_disable_timing` command.

- `library_disable`

  Delay arc disabled due to library disables. These disables may be caused by the following conditions:

  - Disables specified using the `set_disable_timing` command.
  - The arc is an asynchronous arc and the global `lib_build_asynch_arc` is set to `false.`
  - The arc is a conditional default timing arc and the global `lib_build_timing_cond_default_arc` is set to `false`.

- `snipped`

  Loop snipped arcs — arcs automatically disabled by the Encounter software as part of a feedback loop.

`-view` *`viewName`*	Generates the report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Example

- The following command reports all disabled arcs in the design:

  ```
  report_inactive_arcs Report
  ```

  The following example shows the report generated:

Example 32-11 report_inactive_arcs Report

```
From             To                  Summary             CheckType   Reason
-------------------------------------------------------------------------------
U1/U2/reg[0]/E ^ U1/U2/reg[0]/D ^    const                   U1/U2/reg[0]/D = 0
U1/U2/reg[0]/E ^ U1/U2/reg[0]/D v    const                   U1/U2/reg[0]/D = 0
U1/RAM1/A ^      U1/RAM1/D v         const                       mode (RW) = 0
M0/M1/D ^        M0/M1/Q ^           const                       (EN) == 0
A0/D v            A0/Q ^             disable                     User Disable
N0/A v           N0/Q ^              snipped                        loop arc
DREG1/CP ^       DREG1/D ^           const setup                    DREG1/D = 1
DREG1/CP ^       DREG1/D V           const Setup                    DREG1/D = 1
N0/A ^           N0/B ^              disable_clock_gating   setup    N0/A = 1
N0/A v           N0/B ^              disable_clock_gating   setup    N0/A = 1
D0/A ^           D0/Z v              library_disable             library disable
```

- When the software is in multi-mode multi-corner analysis mode, the report generated lists the disabled arcs by active analysis view, as shown in the following example:

Example 32-12 report_inactive_arcs MMMC Analysis Mode

```
-------------------------------------------------------------------------------------+
|  Flags : | const                | propagated constant      |
|          | snipped              | loop snipped arcs        |
|          | disable              | set_disable_timing       |
|          | disable_clock_gating | set_disable_clock_gating |
|          | library_disable      | set_disable_cell_timing  |
|          | Missing_Phase        | No Arc Phase Data        |
|-------------------------------------------------------------------------------------|

| From     | To       | DisableType   | ArcType      | Reason        | View

|----------+----------+---------------+--------------+---------------+-------------|

| RS/CP ^  | RS/QN ^  | Missing_Phase | rising_edge  |Missing Phase  | Funct-WCCOM-Cbest|
|          |          |               |              | Data          |                  |

| RS/CP ^  | RS/QN v  | Missing_Phase | rising_edge  |Missing Phase  | Funct-WCCOM-Cbest|
|          |          |               |              | Data          |                  |

| RS/CP ^  | RS/QN ^  | Missing_Phase | rising_edge  |Missing Phase  | Funct-WCCOM-     |
|          |          |               |              | Data          | Cworst           |

| RS/CP ^  | RS/QN v  | Missing_Phase | rising_edge  |Missing Phase  | Funct-WCCOM-     |
|          |          |               |              | Data          | Cworst           |

| RS/CP ^  | RS/QN ^  | Missing_Phase | rising_edge  |Missing Phase  | Funct-WCCOM-     |
|          |          |               |              | Data          | rcWorst          |

| RS/CP ^  | RS/QN v  | Missing_Phase | rising_edge  |Missing Phase  | Funct-WCCOM-     |
|          |          |               |              | Data          | rcWorst          |
```

report_min_pulse_width

```
report_min_pulse_width
     [-verbose]
     [-path_type {summary | short | full_clock}]
     [-pins object_list]
     [-violation_only]
     [-view view_name]
     [ { > | >> } filename | -tcl_list]
```

Reports a set of minimum pulse width violations on pins and ports on the clock network. The report shows the required pulse width, the actual pulse width, and by how much the constraint is violated or met. This is reported separately for sequential elements and instances on the clock tree.

Parameters

`{> | >>} filename`

When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it.

When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

The *filename* parameter must be the last argument in the list. The *filename* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the `.gz` extension to the file name.

Default: Report is displayed on standard output without being saved.

`-path_type {summary | short | full_clock}`

Specifies the type of report to generate.

Default: `summary`

`summary` Generates a summary report for each pulse width check that shows the required width, the actual width, and the resulting slack.

	`short`	Generates a path report for each pulse width check that lists the latency of each edge of the pulse as a single lumped value, instead of reporting a detailed path report.
	`full_clock`	Generates a path report for each pulse width check that shows the detailed pin-to-pin path for each edge of the pulse.
`-pins` *`object_list`*	Specifies a set of ports and pins on the clock network. If this list is not specified, then the report contains all the pins and ports in the current design.	
`-tcl_list`	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together. See Example 32-16 on page 1728 for an example of the `-tcl_list` output.	
`-verbose`	Provides detailed information about minimum pulse width calculations.	
`-view` *`view_name`*	Generates the report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.	
`-violation_only`	Reports only the negative slack minimum pulse width violations.	

Examples

- The following command generates a report of the minimum pulse width violations on pins and ports on the clock network:

  ```
  report_min_pulse_width
  ```

  The software generates the following report:

Example 32-13 report_min_pulse_width Report

```
+-------------------------------------------------------+
|              SEQUENTIAL CLOCK PULSE WIDTH              |
|-------------------------------------------------------|
| Pin                   | Required |  Actual  |   Slack   |
|                       |  Pulse   |  Pulse   |           |
|                       |  Width   |  Width   |           |
|-----------------------+----------+----------+-----------|
| i_sub/ff2/CK (low)    | 5.000000 | 1.868500 | -3.131500 |
| i_sub/ff1_1/CK (low)  | 5.000000 | 1.929800 | -3.070200 |
| i_sub/ff1_1/CK (high) | 5.000000 | 2.070200 | -2.929800 |
| i_sub/ff2/CK (high)   | 5.000000 | 2.131500 | -2.868500 |
| ff0_1/CK (low)        | 1.100000 | 2.000000 |  0.900000 |
| ff0_1/CK (high)       | 0.800000 | 2.000000 |  1.200000 |
+-------------------------------------------------------+
+-------------------------------------------------+
|              CLOCK TREE PULSE WIDTH             |
|-------------------------------------------------|
| Pin             | Required |  Actual  |   Slack   |
|                 |  Pulse   |  Pulse   |           |
|                 |  Width   |  Width   |           |
|-----------------+----------+----------+-----------|
| buf0_2/Y (low)  | 5.000000 | 1.868500 | -3.131500 |
| buf0_1/Y (low)  | 5.000000 | 1.929800 | -3.070200 |
| buf0_2/A (low)  | 5.000000 | 1.929800 | -3.070200 |
| buf0_1/A (high) | 5.000000 | 2.000000 | -3.000000 |
| buf0_1/A (low)  | 5.000000 | 2.000000 | -3.000000 |
| clk1 (high)     | 5.000000 | 2.000000 | -3.000000 |
| clk1 (low)      | 5.000000 | 2.000000 | -3.000000 |
| buf0_1/Y (high) | 5.000000 | 2.070200 | -2.929800 |
| buf0_2/A (high) | 5.000000 | 2.070200 | -2.929800 |
| buf0_2/Y (high) | 5.000000 | 2.131500 | -2.868500 |
+-------------------------------------------------+
```

- The following command generates detailed information about minimum pulse width calculations for the `i_sub/ff1_1/CK` pin:

  ```
  report_min_pulse_width -verbose -pins {i_sub/ff1_1/CK}
  ```

  The software generates the following report:

Example 32-14 report_min_pulse_width -verbose Report

```
Sequential pulse width check
Pin name: i_sub/ff1_1/CK
     +-------------------------------------------------------+
     | Point                     |  Incr  |  Path  |  Edge |
     |---------------------------+--------+--------+-------|
     +---------------------------+--------+--------+-------+
     | clock clk (fall edge)     |  2.000 |  2.000 |       |
     | clk1 (in)                 |  0.000 |  2.000 |   f   |
     | buf0_1/Y (BUFX1)          |  0.790 |  2.790 |   f   |
     | i_sub/ff1_1/CK (DFFHQX1)  |  0.009 |  2.799 |   f   |
     | data arrival time         |        |  2.799 |       |
     |                           |        |        |       |
     | clock clk (rise edge)     |  0.000 |  0.000 |       |
     | clk1 (in)                 |  0.000 |  0.000 |   r   |
     | buf0_1/Y (BUFX1)          |  0.720 |  0.720 |   r   |
     | i_sub/ff1_1/CK (DFFHQX1)  |  0.009 |  0.729 |   r   |
     | user pulse width          | -5.000 | -4.271 |       |
     | phase shift               |  4.000 | -0.271 |       |
     | data required time        |        | -0.271 |       |
     +---------------------------+--------+--------+-------+
     | data required time        |        | -0.271 |       |
     | data arrival time         |        |  2.799 |       |
     +---------------------------+--------+--------+-------+
     | slack (VIOLATED)          |        | -3.070 |       |
     +-------------------------------------------------------+
```

- The following command generates a report that only includes the negative slack violations:

  ```
  report_min_pulse_width -violation_only
  ```

The software generates the following report:

```
+-------------------------------------------------------+
|              SEQUENTIAL CLOCK PULSE WIDTH              |
|-------------------------------------------------------|
| Pin                   | Required |  Actual  |   Slack   |
|                       |  Pulse   |  Pulse   |           |
|                       |  Width   |  Width   |           |
|-----------------------+----------+----------+-----------|
| i_sub/ff2/CK (low)    | 5.000000 | 1.868500 | -3.131500 |
| i_sub/ff1_1/CK (low)  | 5.000000 | 1.929800 | -3.070200 |
| i_sub/ff1_1/CK (high) | 5.000000 | 2.070200 | -2.929800 |
| i_sub/ff2/CK (high)   | 5.000000 | 2.131500 | -2.868500 |
+-------------------------------------------------------+
+-------------------------------------------------+
|              CLOCK TREE PULSE WIDTH             |
|-------------------------------------------------|
| Pin             | Required |  Actual  |   Slack   |
|                 |  Pulse   |  Pulse   |           |
|                 |  Width   |  Width   |           |
|-----------------+----------+----------+-----------|
| buf0_2/Y (low)  | 5.000000 | 1.868500 | -3.131500 |
| buf0_1/Y (low)  | 5.000000 | 1.929800 | -3.070200 |
| buf0_2/A (low)  | 5.000000 | 1.929800 | -3.070200 |
| buf0_1/A (high) | 5.000000 | 2.000000 | -3.000000 |
| buf0_1/A (low)  | 5.000000 | 2.000000 | -3.000000 |
| clk1 (high)     | 5.000000 | 2.000000 | -3.000000 |
| clk1 (low)      | 5.000000 | 2.000000 | -3.000000 |
| buf0_1/Y (high) | 5.000000 | 2.070200 | -2.929800 |
| buf0_2/A (high) | 5.000000 | 2.070200 | -2.929800 |
| buf0_2/Y (high) | 5.000000 | 2.131500 | -2.868500 |
+-------------------------------------------------+
```

- When the software is in multi-mode multi-corner timing analysis mode, you can specify the `-view` parameter to generate a report only for a specific analysis view:

```
report_min_pulse_width -view s2
```

```
+-----------------------------------------------+
|       SEQUENTIAL CLOCK PULSE WIDTH            |      |
|-----------------------------------------------+------|
| Pin           | Required | Actual | Slack | View |
|               |  Pulse   |  Pulse |       | Name |
|               |  Width   |  Width |       |      |
|---------------+----------+--------+-------+------|
| RS/CK (low)   |  0.640   |  5.000 | 4.360 | s2   |
| RT/CK (low)   |  0.640   |  5.000 | 4.360 | s2   |
| RS/CK (high)  |  0.390   |  5.000 | 4.610 | s2   |
| RT/CK (high)  |  0.390   |  5.000 | 4.610 | s2   |
+-----------------------------------------------+
```

Related Commands

report_analysis_coverage

report_timing

set_min_pulse_width

report_mode

```
report_mode
     instance_list
     [-view viewName]
```

Reports the status of library modes associated with a specified instance. The report includes the status of each mode and its condition, for each mode group defined in the timing library.

Parameters

instance_list	Generates a report for the specified instances.
-view *viewName*	Reports the status of library modes in the specified active analysis view. *Default:* Reports the status of library modes in all active analysis views.

Examples

- The following command activates the `tst_mode_disable` library mode for instances of `i_ram`:

  ```
  set_mode tst_mode_disable [get_cells i_ram]
  ```

- The following command sets the active setup and hold analysis views:

  ```
  set_analysis_views -setup {v1 v2} -hold {v3}
  ```

- The following command reports the status of all modes associated with `i_ram`:

  ```
  report_mode i_ram:
  ```

  ```
  +-------------------------------------------------------------+
  |         Mode Group testmode of i_ram                        |
  |-------------------------------------------------------------|
  |    Mode Name      |  Status   |  Condition     |  View Name |
  |-------------------+-----------+----------------+------------|
  | tst_mode_enable   | Inactive  |  TST_BYPASS    |    v1      |
  | tst_mode_disable  | ACTIVE    |  !(TST_BYPASS) |    v2      |
  +-------------------------------------------------------------+
  ```

- The following command reports the status of all modes associated with `i_ram` in the analysis view `v1`:

```
report_mode i_ram -view v1
```

```
+-------------------------------------------------------------+
|         Mode Group testmode of i_ram                        |
|-------------------------------------------------------------|
|    Mode Name      |  Status   |  Condition     |  View Name |
|-------------------+-----------+----------------+------------|
| tst_mode_enable   | Inactive  |  TST_BYPASS    |    v1      |
+-------------------------------------------------------------+
```

report_net

```
report_net
     [-min_fanout int]
     [-max_fanout int]
     {-net list_of_net_name_or_id | -pin list_of_pin_name_or_id}
     [-tcl_list]
     [-hier]
     [[{> | >>} filename | -output filename]
      | -tcl_list]
```

Reports the net information on the current module. The information includes the net name, the number of source pins, sink pins and bidirectional pins on the net, wire capacitance, and total capacitance.

Electrical information such as pin capacitance, and wire capacitance with which the net is associated are also included.

Use the `-pin` option to report the net that is connected to the specified pin (or port). Both `-net` and `-pin` cannot be specified at the same time. The net (or pin) names can be hierarchical names that are unique relative to the top cell.

This command only searches for the defined nets and does not modify any object in database.

Parameters

{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename*, `-output`, and `-tcl_list` parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the `.gz` extension to the file name. *Default:* Report is displayed on standard output without being saved.
`-hier`	Reports all of the nets in the design hierarchy. Without this argument, only the nets in the current level of hierarchy are reported.

`-max_fanout` *`int`*

Reports the net whose number of fanouts is less than or equal to the maximum number specified by *`integer`*.

`-min_fanout` *`int`*

Reports the net whose number of fanouts is greater than or equal to the minimum number specified by *`integer`*.

`-net` *`list_of_net_name_or_id`*

Reports the set of nets on the list of net names. This argument cannot be specified at the same time as `-pin`. If neither `-net` or `-pin` is specified, all nets are listed by default.

Note: This argument also reports the capacitance of the net and the input pins.

`-output` *`filename`*

Dumps the report to the specified file instead of to the standard output. The *`filename`* argument must be the last argument in the list.

This parameter cannot be used simultaneously with the `-tcl_list` option.

`-pin` *`list_of_pin_name_or_id`*

Reports the set of nets to which the pins on the list *`list_of_pin_name_or_id`* are connected. This argument cannot be specified at the same time as `-net`.

`-tcl_list` Produces the report in Tcl list format instead of a tabular format. This is useful for extracting information from the report in a Tcl program.

The `-tcl_list`, *`filename`* and `-output` parameters are mutually exclusive; you cannot specify them together.

See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

Examples

- Control the `report_net` table fields, such as setting the minimum and maximum column width using the `set_table_style` command. The table names are `report_net_source_table`, `report_net_sink_table`, `report_net_summary_table`, `report_net`, and `report_net_header`. For

example, the following command sets an indentation of six columns for the table and disables printing of the first column:

```
set_table_style -name report_net_source_table -indent 6 -max_widths {0,}
report_net
```

- The format of the `report_net` report is the same for all the examples, as shown in the first example. The following command reports information about net named `in`, as shown in the following example:

Example 32-15 report_net -net in Report

```
Net Name                  : in
Number of Sources         : 1
Number of Sinks           : 2
Number of Bidis           : 0

Source of parasitics      : none
Net Capacitance           : 0.00
Total Capacitance         : 0.02
+-------------------------------------------------------------------------------------------+
|                                        Source                                             |
|-------------------------------------------------------------------------------------------|
|             Pin             |  Reduced Net |   Driver Load   |     Slew      |       |
|-----------------------------+--------------+-----------------+---------------+-------|
|  Name |  Dir  |  Cell |  Cap  |    Model     |  Model | Ctotal |  Rise |  Fall | Phase |
|-------+-------+-------+-------+--------------+--------+--------+-------+-------+-------|
|  in   |  IN   |  top  |  0.00 | Elmore Delay | CTOTAL |  0.02  |  0.05 |  0.04 |@(D)(P)|
+-------------------------------------------------------------------------------------------+
+-------------------------------------------------------------------------------------------+
|                                    Sinks for pin in                                       |
|-------------------------------------------------------------------------------------------|
|                  Pin                | Reduced Ne |     Delay   |        Slew          |
|-------------------------------------+------------+-------------+----------------------|
|      Name       | Dir |  Cell |  Cap | Parameters | Rise | Fall | Rise | Fall | Phase  |
|-----------------+-----+-------+------+------------+------+------+------+------+--------|
| I_block/B_reg/D | IN  | FD1QA | 0.01 |    0.00    | 0.00 | 0.00 | 0.05 | 0.04 | @(D)(P)|
| I_block/A_reg/D | IN  | FD1QA | 0.01 |    0.00    | 0.00 | 0.00 | 0.05 | 0.04 | @(D)(P)|
+-------------------------------------------------------------------------------------------+
```

- The `-tcl_list`, shown in the following example has two types of structures: named lists and tables. The `report_net` command has one of each. The top structure is a named list, with the structure being `net`. Look at the column you want in the column structure, then use that to index into the row. Columns may change, such as additions and reorders, but the data should still be available.

Example 32-16 -tcl_list Output

```
{ net { <net-header>  <src-desc> <src-desc> ... } }
<net-header> = { net_header { <net-hdr-item> <net-hdr-item> ... } }
<net-hdr-item> = { <item> { <value } }
where <item> is something like net_name, num_srcs, and so on
<src-desc> = { src_%d { <src-table> <sink-table> } }
<src-table> = { src_table { <columns> <row> <row> ... } }
<sink-table> = { sink_table { <columns> <row> <row> ... } }
<columns> = { columns { { column1-name } { column2-name } ... } }
<row> = { row_%d { {column1-value} { columns2-value} ... } }
```

- When the software is in multi-mode multi-corner analysis mode, the `report_net` command reports net information for each analysis view. The view name is appended to the Phase information in the report, as shown in the following example:

Example 32-17 report_net -pin MPY_32_INST/M16X16_INST/i_243/B

```
Net Name                : MPY_32_INST/M16X16_INST/FE_OFN220_ab_a_5_
Number of Sources       : 1
Number of Sinks         : 15
Number of Bidis         : 0
Source of parasitics    : None
Net Capacitance         :   0.084
Total Capacitance       :   0.119

 ------------------------------------------------------------------------------------------+
 SOURCE                                                                                    |
 ------------------------------------------------------------------------------------------|
 | Driver Load             | Slew                    |                                   |
 +-------------------------+-------------------------+-----------------------------------|
 | Model     | Ctotal      | Rise        | Fall      |   Phase                           |
 +-----------+-------------+-------------+-----------+-----------------------------------|
 | CTOTAL    |  1.000      | 0.097       | 0.061     | vclk3(D)(P)(core+best-rcTyp)*     |
 | CTOTAL    |  1.000      | 0.136       | 0.080     | vclk1(D)(P)(io+typ-rcMax)*        |
 | OTAL      |  1.000      | 0.161       | 0.100     | vclk3(D)(P)(core+worst-rcTyp)*    |
 | CTOTAL    |  1.000      | 0.097       | 0.061     | vclk3(D)(P)(core+best-rcTyp)*     |
 | CTOTAL    |  1.000      | 0.136       | 0.080     | vclk1(D)(P)(io+typ-rcMax)*        |
 | CTOTAL    |  1.000      | 0.161       | 0.100     | vclk3(D)(P)(core+worst-rcTyp)*    |
```

- The following command reports the net to which the pin `interesting_pin` is connected:

  ```
  report_net -pin interesting_pin
  ```

- The following command reports all the nets in the current module:

  ```
  report_net -net *
  ```

- The following command reports all nets with a fanout greater than 16:

  ```
  report_net -net * -min_fanout 16
  ```

Related Information

`report_path_exceptions`

`report_timing`

report_path_exceptions

```
report_path_exceptions
     [-early | -late | -both]
     [-ignored]
     [-view view_name]
     [{> | >>} filename | -tcl_list]
```

Generates a report about path exceptions specified using the set_false_path, set_min_delay, set_max_delay and set_multicycle_path commands.

By default, the report_path_exceptions command only reports activated path exceptions. To report only the path exceptions that have been ignored by timing analysis, specify the -ignored parameter.

A path can have more than one path exception. Multiple path exceptions that match a given path are prioritized. The adjustment on the path is from the path exception with the highest priority. Table 32-1 on page 1549 ranks path exception priorities from highest to lowest.

When the software is in multi-mode multi-corner timing analysis mode, the report generates path exception information for every active analysis view.

Parameters

{> | >>} *filename*

When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it.

When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the .gz extension to the file name.

Default: Report is displayed on standard output without being saved.

-early | -late | -both

Generates the path exception report for either early paths, late paths, or both early and late paths.

The `-both` parameter can be used only when the timing system is in simultaneous setup and hold analysis mode (`timing_enable_simultaneous_setup_hold_mode`).

Default: Generates a report based on the current timing analysis mode.

`-ignored`

Reports only the path exceptions that have been ignored by timing analysis. This parameter can be used for debugging constraints that may be ineffective or incorrectly entered. A path exception may be ineffective if it is covered by, or overridden by, another path exception. For example:

```
set_false_path -to OUT
set_multicycle_path -to OUT 2
```

In the above situation, the `set_multicycle_path` command would be ignored because the false path has a higher priority.

`-tcl_list`

Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation.

The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

`-view` *`view_name`*

Generates a path exception report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- The following commands shows that a false path has priority over cycle addition on the same path:

```
set_false_path -from I_block/A_reg/Q -to J_block/D_reg/D
set_multicycle_path -from I_block/A_reg/Q -to J_block/D_reg/D 2
```

```
report_path_exceptions -early
```

```
+-------------------------------------------------+
|      From       |       To        |     Early     |
|-----------------+-----------------+---------------|
| I_block/A_reg/Q | J_block/D_reg/D | false         |
| I_block/A_reg/Q | J_block/D_reg/D |               |
+-------------------------------------------------+
```

The term `false` means that the path is treated as a false path due to the path exception, and `add 2 ignored` means that there was a multicycle path constraint but that it is ignored. This is because the false path takes precedence over the multicycle path. If you notice `ignored` path exceptions in your design, check why they are ignored.

- The following command shows how edges are handled:

```
set_false_path -from_rise {in[0]} -through {out[0]}
set_false_path -from_rise {in[0] in[1]} -to_fall {out[0]}}
set_false_path -from_rise {in[0]}
report_path_exceptions -early
```

```
+-----------------------------------------------------+
|     From      | Through |    To    |     Early     |
|---------------+---------+----------+---------------|
| ^ in[0]       | out[0]  |          | false ignored |
| ^ in[0] in[1] |         | v out[0] | false ignored |
| ^ in[0]       |         |          | false         |
+-----------------------------------------------------+
```

```
report_path_exceptions -late
```

```
+-----------------------------------------------------+
|     From      | Through |    To    |     Late      |
|---------------+---------+----------+---------------|
| ^ in[0]       | out[0]  |          | false ignored |
| ^ in[0] in[1] |         | v out[0] | false ignored |
| ^ in[0]       |         |          | false         |
+-----------------------------------------------------+
```

- When the software is in multi-mode multi-corner timing analysis mode, the software generates path exception information for every active analysis view:

```
+--------------------------------------+
|  From |  To   |     Late      |  View |
|       |       |               |  Name |
|-------+-------+---------------+-------|
| RS/CK |       | add 2         | s2    |
| RS/CK | RT/D  | add 2         | s1    |
| RS/CK |       | add 2 ignored | h3    |
+--------------------------------------+
```

- Specify the `-view` parameter to generate a report only for a specific active analysis view:

```
report_path_exceptions -view s1
```

```
+------------------------------+
|  From |  To   |  Late |  View |
|       |       |       |  Name |
|-------+-------+-------+-------|
| RS/CK | RT/D  | add 2 | s1    |
+------------------------------+
```

Related Information

`report_timing`

report_path_groups

```
report_path_groups
     [-name group_name]
     [{> | >>} filename | -tcl_list]
```

Lists the names of all path groups with the corresponding paths.

Parameters

{> | >>} *filename*	When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it. When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file. The *filename* parameter must be the last argument in the list. The *filename* and -tcl_list parameters are mutually exclusive; you cannot specify them together. **Note:** To write a compressed report, add the .gz extension to the file name. *Default:* Report is displayed on standard output without being saved.
-name *group_name*	Limits the report to the specified path group. *Default*: All path groups are reported.
-tcl_list	Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation. The -tcl_list and *filename* parameters are mutually exclusive; you cannot specify them together. See Example 32-16 on page 1728 for an example of the -tcl_list output.

Example

The following command shows a sample report:

```
report_path_groups
+------------------------+
|  From  |  To   | Group |
|        |       |  Name |
|--------+-------+-------|
| "CLKA" | "*"   | GRPA  |
+------------------------+
```

report_ports

```
report_ports
     [-type {[input] | [source_insertion] | [insertion] | [clock_root]
     | [uncertainty] | [arrival] | [required] | [external] | [clk_arrival]
     | [port_cap] | [fanout_load] | [fanout_load_limit] | [drive_resistance]
     | [drive_cell] | [slew_time] | [slew_limit] | [constant] | external_detail
     | drive_resistance_detail}]
     [-include_pins]
     [-pins port_name_id_list]
     [-view viewName]
     [> filename | -tcl_list]
```

Reports timing constraints on ports. By default, the command reports all timing constraints on all ports of the current module.

The direction (*DIR*) column of the generated report can contain one of the following abbreviations:

- `IN` indicates that the pin is an input pin.
- `OUT` indicates that the pin is an output pin.
- `INTL` indicates that the pin is an internal pin (a pin inside the instance boundary).

The *Clock Name* column of the report can contain the following symbols:

- `D` indicates it is a data signal in the clock domain.
- `C` indicates it is a clock signal in the clock domain.
- `P` indicates it is a positive phase.
- `N` indicates it is a negative phase.
- Asterisk (`*`) indicates there is a timing exception associated with the signal.

  Note: Currently, it is not possible to expand the asterisk to determine which exception is associated with the reported timing phase.

Parameters

`> filename`
: Writes the report to the specified file name. If the file already exists, the software overwrites it.

  The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

  Note: To write a compressed report, add the `.gz` extension to the file name.

  Default: Report is displayed on standard output without being saved.

`-include_pins`
: Reports timing constraints on instance pins, as well as on ports.

  Default: Reports timing constraints on ports only.

`-pins port_name_id_list`
: Specifies the pins or ports to be reported. Pins can be specified by name, or by object ID in a Tcl list.

`-tcl_list`
: Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation.

  The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

  See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

`-type {assertion_type | assertion_type_list}`

Specifies the constraints to be reported.

Default: All constraints are reported. The following are valid *assertion_type* values:

- `arrival`

  Reports arrival time constraints as set by `set_input_delay` command.

- `clk_arrival`

  Reports clock arrival time constraints as set by the `set_clock_latency` command.

- `clock_root`

  Reports the clock root as set by the `create_clock` command.

- `constant`

  Reports constant constraints as set by the `set_case_analysis` command.

- `drive_cell`

  Reports drive cell constraints as set by the `set_driving_cell` command.

- `drive_resistance`

  Reports drive resistance constraints as set by the `set_drive` command.

- `drive_resistance_detail`

  Reports detail about the input ports with drive resistance constraints.

- `external`

  Reports external delay constraints as set by the `set_output_delay` command.

- `external_detail`

  Reports detail about the output ports with external delay constraints.

- `fanout_load`

  Reports fanout load constraints as set by the `set_fanout_load` command.

- `fanout_load_limit`

  Reports fanout load limit constraints as set by the `set_max_fanout` command.

- `input`

  Reports the input delay constraints as set by the `set_input_delay` command.

- `insertion`

  Reports the network insertion delay as set by the `set_clock_latency` command.

- `port_cap`

  Reports port capacitance constraints as set by the `set_load` command.

- `required`

  Reports required time constraints as set by the `set_output_delay` command.

- `slew_limit`

  Reports slew time limit constraints as set by the `set_max_transition` command.

- `slew_time`

  Reports slew time constraints as set by the `set_input_transition` command.

- `source_insertion`

  Reports the source insertion delay as set by the `set_clock_latency -source` command.

- `uncertainty`

  Reports the clock uncertainty as set by the `set_clock_uncertainty` command.

`-view` *`viewName`* — Generates the report for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- The following command shows a port report, as shown in the following example:

Example 32-18 report_ports Report

				Early		Late	
Pin Name	Dir	Assertion	Clock Name	Rise	Fall	Rise	Fall
I_block/u000/A	IN	slew_time	CLK1(C)(P)	1.60	1.60	1.60	1.60
J_block/D_reg/D	IN	arrival	CLK1(D)(P)		0.07		
J_block/D_reg/D	IN	drive_resistance	CLK1(D)(P)	1.20	1.20	1.20	1.20
clkA	IN	clk_arrival	CLK2(C)(N)	2.00	0.00	2.00	0.00
clkA	IN	slew_time	CLK1(C)(P)	1.30	1.30	1.30	1.30
clkB	IN	clk_arrival	CLK2(C)(P)	0.00	2.00	0.00	2.00
clkB	IN	slew_time	CLK1(C)(P)	1.30	1.30	1.30	1.30
clkC	IN	clk_arrival	CLK2(C)(P)	0.05	0.07	0.05	0.07
clkD	IN	clk_arrival	CLK2(C)(N)	3.00	0.50	3.00	0.50
in	IN	drive_cell	*(D)(P)	BUFA	BUFA	BUFA	BUFA
out	OUT	external	CLK1(C)(N)			1.00	1.00

Pin Name	Dir	Assertion	Value
out	OUT	port_cap	(3.20 : 3.20 : 3.2)
out	OUT	port_cap_limit	4.00
in	IN	slew_limit	1.50
in	IN	port_cap_limit	2.00
clkA	IN	port_cap_limit	2.00
clkB	IN	port_cap_limit	2.00
clkC	IN	port_cap_limit	2.00
clkD	IN	port_cap_limit	2.00

- The following command shows, from a different design, that `input`, `source_insertion` and `clock_root` assertion type options report the constraints in the same format as `arrival`, shown in the following example:

Example 32-19 report_ports -type clock_root source_insertion input -pin clk in

				Early		Late	
Pin Name	Dir	Assertion	Clock Name	Rise	Fall	Rise	Fall
clk	IN	clock_root	CLK(C)(P)				
clk	IN	source_insertion	*(D)(P)	2.50	2.50	2.50	2.50
in	IN	input	CLK(D)(P)	3.00	3.00	3.00	3.00

- The following command shows that the insertion and uncertainty are reported as one value in the following format:

```
insertion              : RISEmin FALLmin : RISEtyp FALLtyp : RISEmax FALLmax
```

```
uncertainty             : EARLYrise EARLYfall LATErise LATEfall
```

- The following command shows that the `-to` uncertainty is denoted by `(T)` next to the number. If a value is not specified, it is denoted by a dash (`-`):

```
report_port -type insertion uncertainty -pin clk
        +-------------------------------------------------------------------------+
        |  Pin  |  Dir  |  Assertion           |             Value                |
        |  Name |       |                      |                                  |
        |-------+-------+----------------------+----------------------------------|
        |  clk  |  IN   |  insertion           | ( 1.50 3.50 : - - : 2.50 4.50 ) |
        |  clk  |  IN   |  uncertainty         |         1.00(T) 1.10 - 1.20(T) |
        +-------------------------------------------------------------------------+
```

Related Information

report_clocks

report_net

report_timing

report_timing

```
report_timing
     [-clock_from clk_signame_list [-edge_from {lead | trail}]]
     [-clock_to clk_signame_list] [-edge_to {lead | trail}]]
     [-rise | -fall]
     [-early | -late]
     [ -check_type {setup | hold | pulse_width | clock_period | clock_gating_setup
                  | clock_gating_hold | clock_gating_pulse_width | data_setup
                  | data_hold | recovery | removal | clock_separation
                  | skew | no_change_setup | no_change_hold}
     [ -max_paths integer | -max_points npoint [-nworst integer]
      | -begin_end_pair]
     [{-from | -from_rise | -from_fall} pin_list]
     [{-through | -through_rise | -through_fall} pin_list]
     [{-to | -to_rise | -to_fall} pin_list]
     [-point_to_point]
     [-check_clocks]
     [-path_group groupname_list]
     [-net]
     [-unique_pins]
     [-path_type {end | summary | full | full_clock | end_slack_only
                  | summary_slack_only}]
     [-max_slack float]
     [-min_slack float]
      | -unconstrained [-delay_limit float]
     ]
     [-view {viewName}]
     [-format column_list]
     [-sequential aggregatedResultFileName]
     [-collection]
     [-retime {locv | ssta |path_slew_propagation}]
     [-machine_readable | -tcl_list]
     [-derate_summary]
     [-worst_rc_corner]
     [-sort_slack_by {ssta_yield | ssta_violation | ssta_NSigma  |
                  ssta_path_criticality}]
     [-ssta_jpdf]
     [-ssta_criticality N]
     [-ssta_percentile N]
     [-ssta_sigma_multiplier N]
     [{> | >>} filename]
```

Generates a timing report that provides information about the various paths in the design.

You can use the `-format` parameter to customize the reports to your needs by requesting the exact fields in which you have an interest. Valid format columns are: `adjustment`, `annotation`, `arc`, `arrival`, `cell`, `delay`, `direction`, `edge`, `fanin`, `fanout`, `incr_delay`, `instance`, `instance_location`, `load`, `locv_derate`, `net`, `phase`, `pin`,

`hpin`, `pin_location`, `required`, `retime_delay`, `retime_slew`, `slew`, `stolen`, and `user_derate`.

You can also use a combination of the `-format` and `-tcl_list` parameters to integrate the timing reports into your Tcl scripts. You can use the `-from` parameter to limit the number of paths reported, and to find specific paths in the design.

Reports typically contain data on the delay through the entire path. The start node and the end node of each path is identified.

The timing report contains the following information:

- The slack times of the arriving signal at the end node
- The start node
- The associated transitions
- The signal required times and the actual signal arrival times
- Any phase shifts applied when evaluating timing checks
- Any CPPR values applied to timing check evaluation

Parameters

`{> | >>}` *`filename`*
When specified with >, writes the report to the specified file name. If the file already exists, the software overwrites it.

When specified with >>, writes the report to the specified file name. If the file already exists, the software concatenates the report to the end of the file.

The *`filename`* parameter must be the last argument in the list. The *`filename`* and `-tcl_list` parameters are mutually exclusive; you cannot specify them together.

Note: To write a compressed report, add the `.gz` extension to the file name.

Default: Report is displayed on standard output without being saved.

`-begin_end_pair`
Reports all violating paths between unique source and target register pairs. To report other paths, specify this parameter with the `-max_slack` parameter.

Paths are not sorted by slack; instead, paths with the same source pins are reported adjacent to each other.

When specified with the `-from`, `-through`, or `-to` parameters, the software only reports the unique register pair paths that match the path description.

Note: You cannot specify this parameter with the `-max_paths`, `-nworst`, `-max_points`, `-unique_pins`, or `-min_slack` parameters.

`-check_clocks`
Generates reports based on timing paths on the clock network instead of the standard timing to data endpoints. This report includes clock paths that end at the reference end of a check or a clock gating end point.

Default: Data paths and all the other clock paths (for example, a clock path ending at a D pin of a register) are reported.

`-check_type` *`check_type`*

Reports only the paths that end at the specified timing check.

The following check types can be reported when the software is in setup analysis mode: `setup`, `pulse_width`, `clock_period`, `clock_gating_setup`, `clock_gating_pulse_width`, `data_setup`, `recovery`, `clock_separation`, and `no_change_setup`.

The following check types can be reported when the software is in hold analysis mode: `hold`, `clock_gating_hold`, `data_hold`, `removal`, `skew`, and `no_change_hold`.

Do not use this parameter with the `-unconstrained`, `-early`, or `-late` parameters.

`-clock_from` *`clk_signame_list`*

Generates reports based on source clock waveform(s). Reports only those paths whose source clocks are the clock signals in *`clk_signame_list`*.

`-clock_to` *`clk_signame_list`*

Generates reports based on target clock waveform(s). Reports only those paths whose target clocks are the clock signals in *`clk_signame_list`*.

`-collection` Returns a collection of timing paths. This is useful for performing Tcl queries on selected timing reports.

`-delay_limit` *`float`* Specifies the path delay limit for unconstrained paths (`-unconstrained` option).

For early paths (`-early` option), reports only those paths with path delay less than the delay limit. For late paths (`-late` option) reports only those paths with path delay more than the delay limit.

Note: The `-delay_limit` option only can be used in conjunction with the `-unconstrained` option.

`-derate_summary` Generates an LOCV derating summary table for the launch and capture paths, in addition to the timing report.

`-early | -late`

Generates the timing report for early paths (hold checks) or late paths (setup checks).

Default: `-late`

Note: If you specify a check, setup or hold, using the `setAnalysisMode` command, the `report_timing` command reports that check by default.

`-edge_from {lead | trail}`

Generates reports based on source clock edge, either leading or trailing.

Note: The `-edge_from` parameter only can be used in conjunction with the `-clock_from` parameter.

Default: Generates reports based on both source clock edges

`-edge_to {lead | trail}`

Generates reports based on the target clock edge, either leading or trailing.

Note: The `-edge_to` parameter only can be used in conjunction with the `-clock_to` parameter.

Default: Generates reports based on both target clock edges

`-format` *`column_list`*

Formats the report according to the *`column_list`*. The *`column_list`* specifies which columns to display in the timing report and the order in which they appear. For example:

```
-format {hpin cell delay required arrival
required edge}
```

See Table 32-7 on page 1757 for a list of valid options.

Default: {`instance arc cell delay arrival required`}

The default net format (with `-net` option) for the full path is: `{hpin edge net cell delay arrival required}`.

If the `-unconstrained` option is specified, the `required` column is not displayed.

The `-format` option cannot be used with the `-path_type end` or `-path_type summary` options.

`-from | -from_rise | -from_fall` *`pin_list`*

Reports paths starting from the pin(s) specified by the *`pin_list`*. Using `-from_rise` (or `-from_fall`) specifies that the rising (or falling) edge of the signals on the pins in *`pin_list`* are the start of the paths.

Note: Use this option with the `-through` and `-to` options for specifying particular paths in the design.

`-machine_readable`

Generates detailed timing report in machine-readable format. This report is used for debugging timing results using the timing debug feature.

For more information on debugging timing results, see "Debugging Timing Results" in Encounter User Guide.

`-max_paths` *`integer`*

Reports the specified number of worst paths in the design, regardless of the endpoint. This is useful, but can be time consuming if a large number of paths are requested. The paths are always sorted based on slack.

Default: worst path

The `-max_paths` option cannot be used with the `-max_points` option.

`-max_points` *`integer`*

Reports the worst path it finds to each endpoint up to the number specified by the `-max_points` option. If `-path_type end` option is specified, it reports worst path to each endpoint. This is the most frequently used report.

Default: Only the worst path to one endpoint is reported.

The `-max_points` option cannot be used with the `-max_paths` option.

`-max_slack` *`float`*

Reports only those paths with slack equal to or less than the value of *`float`* are reported. The `-max_slack` option limits the report to paths that fall into the specified range. A positive slack value indicates that timing was met. A negative value for slack indicates a timing violation.

The `-max_slack` option cannot be used with the `-unconstrained` option.

Typically, you can report all violating endpoints by using: `-max_slack 0.0 -max_points 1000`

Note: This still limits the report to 1000 endpoints. The endpoint limitation can be adjusted to a reasonable number.

`-min_slack` *`float`*

Reports only those paths whose slack is greater than the value of *`float`*.

The `-min_slack` option cannot be used with the `-unconstrained` option.

Generate a timing report containing any path with greater than the specified slack by using the `-max_paths` option. For example, to report all paths with slack greater than 2ns: `-min_slack 2.0 -max_paths 1000`

Note: This still limits the report to 1000 paths. The path limitation can be adjusted to a reasonable number.

`-net`

Adds a row for the net arc. This parameter also separates the cell delay from the wire delay.

Default: The net arc is not shown, and the net delay is added to the following delay.

Note: The `report_timing -net` command displays the net delays separate from the cell delays. However, the net delay is sometimes shown as 0.0.To get the net delays to show up, increase the report precision using the following global:

```
set_global report_precision 5
```

`-nworst` *`integer`*

Specifies the number of paths to be enumerated for each endpoint. Use the `-nworst` option to report all the checks at an end point or use the `-check_type` option to report a specific check.

Default: Only the worst path to each endpoint is reported.

The `-nworst` option cannot be used with the `-max_paths` option.

`-path_group` *`groupname_list`*

Reports only paths contained in the groups specified in *`groupname_list`*.

Report paths belonging to the default path group by using `-path_group default`. The paths that do not belong to any user-specified path group belong to the default path group named `default`.

`-path_type {end | summary | full | full_clock | end_slack_only | summary_slack_only}`

The `path_type` option lets you choose the format of the report by path type. The default format, if the `-path_type` option is not specified is `full`.

Choose one of the following:

`end` — Generates an end point report for each path consisting of an endpoint, cause, slack, arrival time, required time, and phase. This option generates a very fast report. The `-format` option does not have any impact on this report format.

Note: The `-path_type end` format is identical to the format produced by the obsolete `-summary` option.

`summary` — Generates a summary report for each path consisting of a start point, endpoint, cause, slack, arrival time, required time, and phase. The `-format` option does not have any impact on this report format.

`full` Generates a report that displays the full path with accompanying required time and slack calculation. This is the default path type. Control the report format using the `-format` option.

`full_clock` If the path reported ends at a timing check, this option also reports the full clock path (`Other End Path`) in addition to the full data path (`Timing Path`).

`end_slack_only`

Generates an end point report, similar to `-path_type end` report, for each path consisting of an endpoint, slack and cause (as Violated/Met). This option generates a very fast report. The `-format` option does not have any impact on this report format.

Note: The `-path_type end_slack_only` format is identical with the format produced by the obsolete `-summary` parameter.

`summary_slack_only`

Generates a summary report, similar to `-path_type summary`, for each path consisting of a start point, endpoint, slack and cause (as Violated/Met). The `-format` option does not have any impact on this report format.

`-point_to_point`

Traces the worst delay path between a pair of from-to pins. Paths reported with this parameter only include the cumulative delay of the path.

You must specify either the `-from` or `-to` parameter, to define the begin and end points of the path.

If you do not specify a `-from` pin, the standard definition of the begin point of a path applies. If you specify a `-from` pin, it is treated as the begin point of the path. If you specify a `-from` pin that is not a valid begin point, the arrival time of the first pin starts from 0.

If you do not specify a `-to` pin, the standard definition of the end point applies. If you specify a `-to` pin, it is treated as the end point, and only those paths ending in `-to` pins are reported. If you specify more than one `-to` pin, the worst path to each pin-to-pin is reported.

When from-to pins are specified, the software will trace across the trigger arcs and report a path, if there is a clock reaching the CK pin of the trigger arc. However, if there is a begin or end point present between the `-from` and `-to` pins, the software will not trace beyond it.

If there is no clock reaching the CK pin of the trigger arc, the software cannot trace across the trigger arc, and no path is reported. Additionally, clock path delay is included only when the clock is in propagated mode.

The `-point_to_point` parameter also can be used with the `-max_paths`, `-max_points`, and `-nworst` parameters.

See `report_timing -point_to_point` Sample Reports on page 1766.

`-retime {locv | ssta |path_slew_propagation}`

Reanalyzes the specified set of paths using the specified method.

Note: This feature is not meant to be run on all the paths of a design.

`locv` Takes the most critical paths and reanalyzes them using the LOCV derating factors.

Note: The following `report_timing` parameters are not supported with `-retime locv`: `-unique_pins`, `-unconstrained`, and `-delay_limit`.

`ssta` Reports timing analysis results based on path-based statistical static timing analysis (SSTA). By default, the software runs block-based SSTA.

`path_slew_propagation`

Takes the paths with the worst-case slew propagation and retimes them one at a time, using the actual slews for the path. This results in a more precise calculation of the actual delays and slack.

The default timing analysis uses worst-case slew propagation to calculate the delays at each stage. As a result, timing analysis results can be pessimistic.

Note: This feature should be run only on paths that are not meeting the slack margin criteria, to help identify actual failing paths from those that are simply failing due to the conservative delay calculation.

`-rise | -fall` Reports the path with the specified edge on the endpoint.

If an endpoint is specified using `-to_rise` (or `-to_fall`) option, the `-rise` (or `-fall`) option is ignored and paths with edge specified by `-to_rise` (or `-to_fall`) are reported.

`-sequential` *`aggregatedResultFile`*

Analyzes all active analysis views one at a time, then generates the specified aggregated result file. The result file indicates with which view each of the critical paths reported is associated.

During analysis, the software generates a machine readable (`.mtarpt`) report file for each analysis view. The software names these intermediate reports by the type of analysis and the view name. For example, `setup_v1.mtarpt` and `hold_v4.mtarpt`.

When the software finishes analyzing all of the views, the results are aggregated and output to the specified file in `.mtarpt` format. You can then use Timing Debug to analyze the results.For more information, see "Debugging Timing Results," in the *Encounter User Guide*.

Note: Parameter `-sequential` is obsolete and is no longer recommended.

`-sort_slack_by {ssta_yield | ssta_violation | ssta_NSigma | ssta_path_criticality}`

Sorts the SSTA report in the specified order.

`-ssta_criticality` *`N`* Reports the top *`N`* pins with their criticalities and slack values at (mean-3sigma). Criticality of a pin is the probability of that pin on the critical path.

`-ssta_jpdf` Reports joint probability density function (JPDF) of all the paths in addition to individual path reporting. By default, JPDF is not reported.

`-ssta_percentile` *`N`* Specifies the yield target for SSTA reports. Use this option to trade-off between run time and accuracy. By default, SSTA uses yield value equivalent to 99.86%.

`-ssta_sigma_multiplier` *`N`*

Reports *`N`* sigma value (Mean+N*sigma) of the slack.

Default: `-3`

`-tcl_list`

Produces the report in Tcl list format instead of a tabular format. This is useful for integrating timing with custom Tcl functions, and also for customizing report generation.

The `-tcl_list` and *`filename`* parameters are mutually exclusive; you cannot specify them together.

See Example 32-16 on page 1728 for an example of the `-tcl_list` output.

`-through | -through_rise | -through_fall` *`pin_list`*

Reports paths that pass through the pin(s) specified by the *`pin_list`*. Any number of `-through` pins can be specified. Using `-through_rise` (or `-through_fall`) specifies that the paths go through the rising (or falling) edge of the signals on the pins in *`pin_list`*.
The *`pin_list`* is a logical OR function. The resulting path may pass through any of the pins in the `-through` *`pin_list`*. To force the report to pass through multiple pins, separate `-through` statements are needed.

`-to | -to_rise | -to_fall` *`pin_list`*

Reports paths leading to the pin(s) specified by the *`pin_list`*. Pins in the *`pin_list`* can be either pins on the design boundary (ports) or pins on an instance. Only one list of `-to` pins can be specified per report. Using `-to_rise` (or `-to_fall`) specifies that the rising (or falling) edge of the signals on the pins in *`pin_list`* are at the end of the paths.

`-unconstrained`

Reports only the unconstrained paths (paths with no slack). If no paths are found, there may be constrained paths or false paths or the path may not exist. Each signal arriving at the path end node which does not have a matching required time, results in an unconstrained path.

The `report_timing` command without the `-unconstrained` option reports only constrained paths. If no constrained path is found, there may be unconstrained paths or false paths or the path may not exist. If no constrained or unconstrained path is found, the path is either false or does not exist structurally.

Note: The `-min_slack` and `-max_slack` options cannot be specified with the `-unconstrained` option. See the `-delay_limit` option described below.

See Figure 32-1 on page 1772 for the conditions under which a path is reported as constrained or unconstrained.

`-unique_pins`

Reports paths through unique set of pins. Only one path (the worst) is reported for unique set of pins. This option suppresses multiple paths that differ only because of transition polarity or conditional arcs.

Note: Using this option may significantly affect runtime and memory usage if the number of paths to be reported is high.

`-view {`*`viewName`*`}`

Generates report for the specified analysis view.

You must be in multi-mode multi-corner analysis mode in order to use this parameter. To create a view, you use the `create_analysis_view` command. To set an active view for analysis, you use the `set_analysis_view` command.

Default: Reports the worst path to each endpoint across all analysis views.

`-worst_rc_corner`

Identifies worst interconnect parameter combination that leads to minimum slack (desired) and generates a report for the corner.

Table 32-7 Report Timing—Column List Options

Option	Description
`adjustment`	Reports generated clock adjustment values. When reporting the detailed path for a generated clock, there are circumstances where the arrival time needs to be adjusted when a create_generated_clock assertion is encountered along the path. The `adjustment` argument can be used to provide a clearer indication of why a “jump” in arrival time is seen in the path report. There are two primary reasons for generated clock-related adjustment to occur. ■ If the `timing_enable_genclk_edge_based_source_latency` global variable is set to `false`, it is possible that the edge-to-edge requirement specified by the generated clock cannot be satisfied by an existing logic path. This can happen when the source clock to generated clock polarity of the constraint is non-inverting, but the clock path itself constrains an inverting path. In this case, it would be normal to see a half-cycle of adjustment as the software compensates for the path versus constraint mismatch. If the above mentioned global variable is set to `true`, the software handles the path versus constraint mismatch by zeroing the source latency of the generated clock, and forcing alignment of the source and generated clock. As a result, no further adjustment is required, or reported. ■ In cases where `create_generated_clock -edge_shift` is specified, the adjustment column will show the amount of shift specified.

Table 32-7 Report Timing—Column List Options, *continued*

Option	Description
`annotation`	When `-net` is specified, reports RC parasitic annotations for net arcs, and delay annotations for either gate or net arcs. The following annotations are reported: SDF: SDF back-annotation. When `-net` is not specified, the annotation status reflects the gate arc, not the interconnect arc. SPEF: RC network back-annotation `LumpedRC`: Lumped RC annotation `DlyAssert`: Assertion set with `set_annotated_delay` command <none>: All others
`arc`	Reports the arc as described by the from pin, from pin edge, to pin, and to pin edge. For example, the arc from the rising edge of pin A to the falling edge of pin Z is reported as: `A ^->Z v`
`arrival`	Reports the arrival time on the pin.
`cell`	Reports the cell name of the given pin's instance.
`delay`	Reports arc delay. If stolen slack at the arc output pin (output of transparent latches) is not zero, the `stolen` column is also displayed along with this column.
`delay_sensitivity`	Reports sensitivity of each parameter of arc delay.
`delay_sigma`	Reports standard deviation of each parameter of arc delay.
`direction`	Reports the pin direction (IN, OUT).
`edge`	Reports the edge on the pin (^=rise, v=fall).
`fanin`	Reports the number of source nodes of the net connected to the timing pin.
`fanout`	Reports the number of sinks of the net connected to the timing pin.
`hpin`	Reports the hierarchical name for the given pin.

Table 32-7 Report Timing—Column List Options, *continued*

Option	Description
`incr_delay`	Reports the incremental delay of the pertaining arc. The software reads the incremental delay from an incremental SDF file using the `read_sdf` command, or through Tcl sourcing a file of `set_annotated_delay -net -delta_only` constraints.
`instance`	Reports the hierarchical name of the given pin's instance.
`instance_location`	Reports the location (x,y) of the instance. If the design is not placed, the column is blank. This argument displays the `instance` column automatically.
`load`	Reports the total capacitive load on a given pin.
`locv_derate`	Reports the LOCV derating factor used for the cell or net.
`net`	Reports the hierarchical name of the net connected to given pin.
`phase`	Reports the phase name on pin.
`pin`	Reports the reference name for the given pin.
`pin_location`	Reports the location (x,y) of the pin. If the design is not placed, the column is blank.
`required`	Reports the required time on the pin.
`retime_delay`	Reports the recalculated arc delay based on the path being reanalyzed with the specified retime method.
`retime_delay_sensitivity`	Reports sensitivity of each parameter of arc's retimed delay.
`retime_delay_sigma`	Reports standard deviation of each parameter of arc's retimed delay.
`retime_slew`	Reports the recalculated slew based on the path being reanalyzed with the specified retime method.
`retime_slew_sensitivity`	Reports sensitivity of each parameter of arc's sink pin's retimed slew.

Table 32-7 Report Timing—Column List Options, *continued*

Option	Description
`retime_slew_sigma`	Reports standard deviation of each parameter of arc's sink pin's retimed slew.
`slew_sensitivity`	Reports sensitivity of each parameter of arc's sink pin's slew.
`slew`	Reports the propagated slew at the given pin.
`slew_sigma`	Reports standard deviation of each parameter of arc's sink pin's slew.
`stolen`	Reports the slack stolen (or the time given to previous stage) at the given pin. If the slack is not zero and the delay column is specified, this column is displayed by default. (only visible on the output of transparent latches.)
`user_derate`	Reports the derating scale factors set with the `set_timing_derate` command. **Note:** The timing system cannot provide an accurate derating factor for the lumped gate and wire delay in the following situations: ❑ The `-net` parameter is not specified with `report_timing`. ❑ The `set_timing_derate` factors for `-cell_delay` and `-net_delay` are not equivalent. In these situations, an asterisk (*) is reported in the User Derate column.
`when_cond`	Reports the "when" condition of the arc specified in the library.

Reporting Timing Checks Between Various Types of Clocks

The following table summarizes the timing checks and the min/max delay checks performed based on the types of launching and capturing clocks.

Table 32-8 Timing and Minimum and Maximum Delay Checks Between Clocks

Launching Clock	Capturing Clock	Timing Checks	Path Delay Checks
@	@	NO	CPD
Z	@	NO	CPD
NZ	@	No	CPD
@	Z	YES	SPD
@	NZ	YES	SPD
NZ	Z	NO	SPD
Z	NZ	YES	SPD
Z	Z	YES	SPD
NZ	NZ	YES	SPD

- NZ (Non Zero Period Clock - Regular Clock)
- Z (Zero Period Clock - Regular clock whose period is equal to zero)
- @ (Asynchronous Clock - Clock which triggers the default input delay and the input delay without a clock)
- CPD (Combinational Path Delay)

SPD (Sequential Path Delay)**Examples**

- The following command reports paths leading to the pin(s) specified by the *pin_list*:

```
report_timing -to O

Path 1: VIOLATED External Delay Assertion
Endpoint:   O     (v) checked with  leading edge of 'CK1'
Beginpoint: fd3/Q (v) triggered by  leading edge of 'CK1'
Other End Arrival Time      1.00
- External Delay            4.70
+ Phase Shift               5.00
= Required Time             1.30
- Arrival Time              1.70
= Slack Time               -0.40
     Clock Rise Edge                  1.00
     = Beginpoint Arrival Time        1.00
     +-------------------------------------------------------------+
     | Instance |     Arc     |  Cell | Delay | Arrival | Required |
     |          |             |       |       |  Time   |   Time   |
     |----------+-------------+-------+-------+---------+----------|
     |          | CK ^        |       |       |    1.00 |     0.60 |
     | an2      | B ^ -> Z ^  | AN2   |  0.00 |    1.00 |     0.60 |
     | fd3      | CP ^ -> Q v | FD1   |  0.70 |    1.70 |     1.30 |
     |          | O v         |       |  0.00 |    1.70 |     1.30 |
     +-------------------------------------------------------------+
```

- The following command shows a report similar to the first with the addition of net arc information.

```
report_timing -to O -net

Path 1: VIOLATED External Delay Assertion
Endpoint:   O     (v) checked with  leading edge of 'CK1'
Beginpoint: fd3/Q (v) triggered by  leading edge of 'CK1'
Other End Arrival Time      1.00
- External Delay            4.70
+ Phase Shift               5.00
= Required Time             1.30
- Arrival Time              1.70
= Slack Time               -0.40
     Clock Rise Edge                  1.00
     = Beginpoint Arrival Time        1.00
     +-------------------------------------------------------------+
     |  Pin   |  Edge |  Net  |  Cell | Delay | Arrival | Required |
     |        |       |       |       |       |  Time   |   Time   |
     |--------+-------+-------+-------+-------+---------+----------|
     | CK     |   ^   | CK    |       |       |    1.00 |     0.60 |
     | an2/B  |   ^   | CK    | AN2   |  0.00 |    1.00 |     0.60 |
     | an2/Z  |   ^   | CLK1  | AN2   |  0.00 |    1.00 |     0.60 |
     | fd3/CP |   ^   | CLK1  | FD1   |  0.00 |    1.00 |     0.60 |
     | fd3/Q  |   v   | O     | FD1   |  0.70 |    1.70 |     1.30 |
     | O      |   v   | O     | latch |  0.00 |    1.70 |     1.30 |
     +-------------------------------------------------------------+
```

- The following command displays the worst late path in the design. The format of the report is similar to the first example:

  ```
  report_timing
  ```

- The following command displays the worst late path to each violating endpoint that has a slack less than -1.0:

  ```
  report_timing -max_slack -1.0
  ```

- The following command displays all the late paths that end at port `out[2]` and that have negative slack up to a maximum of 1000 worst paths. If there are more than 1000 paths, only the 1000 worst paths are reported:

  ```
  report_timing -to out[2] -max_paths 1000 -max_slack 0.0
  ```

- The following command reports the worst late path that starts at `in[0]` and ends at `out[1]`:

  ```
  report_timing -from in[0] -to out[1]
  ```

- The following command displays the three worst paths that start at `in[1]` and end at `out[3]`. With reconvergent fanout, more than one path may exist:

  ```
  report_timing -from in[1] -to out[3] -max_paths 3
  ```

- The following command displays the ten worst paths. Only the paths between the specified pins are enumerated.With reconvergent fanout, more than one path may exist:

  ```
  report_timing -from i102/Z -to i123/A -max_paths 10
  ```

- The following command enumerates the worst ten paths through the given two pins, starting at the beginning points in the design and ending at the endpoints, similar to using the `-from` option:

  ```
  report_timing -through i102/Z -through i123/A -max_paths 10
  ```

- The following command reports the ten worst paths through blocks A and C, or blocks B and C. The path only has to satisfy one element in a through list, but all through lists must be satisfied. The example can be thought of as ((A or B) and C):

  ```
  report_timing -through {A/*B/*} -through {C/*} -max_paths 10
  ```

- The following command forces a path through pin A and pin B. Use the syntax shown in the example. Do not use `-through {A B}`:

  ```
  report_timing -through A -through B
  ```

- The following command reports the worst endpoint and the ten worst paths to that endpoint. Similar to `report_timing -max_paths 10 -to out[1]`, if `out[1]` is the worst endpoint:

  ```
  report_timing -max_points 1 -nworst 10
  ```

- The following command displays the clock path through the clock root to Launch flop:

  ```
  report_timing -through aBC_bs/bs_mex1/r2_00_q1_reg_10_/CK -to .... -path_type
  full_clock
  ```

  You will not see the path for the launch clock if you specify the following command because the report_timing command starts reporting from the CK pin:

  ```
  report_timing -from aBC_bs/bs_mex1/r2_00_q1_reg_10_/CK
  ```

  The first command displays the path leading to the `CK` pin.

- The following command shows the `-unconstrained` option:

  ```
  report_timing -unconstrained
  ```

```
Path 1:Endpoint:   O2     (^)
Beginpoint: reg2/Q (v) triggered by trailing edge of 'vclk'
+-------------------------------------------------+
| Instance |     Arc     |  Cell | Delay | Arrival |
|          |             |       |       |  Time   |
|----------+-------------+-------+-------+---------|
|          | clk1 v      |       |       |   10.00 |
| nand1    | B v -> Z ^  | ND2   |  0.17 |   10.17 |
| buf1     | A ^ -> Z ^  | BUF8A |  0.18 |   10.35 |
| reg2     | CP ^ -> Q v | FD1   |  0.74 |   11.09 |
| nand4    | A v -> Z ^  | ND2   |  0.07 |   11.16 |
|          | O2 ^        |       |  0.00 |   11.16 |
+-------------------------------------------------+
```

- The following command shows the `-unconstrained -path_type end` report:

```
report_timing -unconstrained -path_type end
```

```
+-----------------------------------------------------------+
|       Pin | Cause              | Arrival |         Phase |
|-----------+--------------------+---------+---------------|
|      O2 ^ | Unconstrained Path |   10.82 |        vclk N |
|      O1 v | Unconstrained Path |    0.70 |        vclk P |
|  reg2/D ^ | Unconstrained Path |    0.24 | Unconstrained |
|  reg3/D ^ | Unconstrained Path |    0.17 | Unconstrained |
|  reg1/D v | Unconstrained Path |    0.15 | Unconstrained |
|  reg4/D v | Unconstrained Path |    0.15 | Unconstrained |
| nand1/A v | Unconstrained Path |    0.00 | Unconstrained |
+-----------------------------------------------------------+
```

- The following command displays the timing report for analysis view `view2`:

```
report_timing -view view2
Path 1: MET Setup Check with Pin TR2/BR1/CK
Endpoint:   TR2/BR1/D (v) checked with  leading edge of 'WAVE'
Beginpoint: BLK/BR2/Q (v) triggered by  leading edge of 'WAVE'
Analysis View: view2
Other End Arrival Time          0.267
- Setup                         0.298
+ Phase Shift                   2.000
= Required Time                 1.970
- Arrival Time                  1.056
= Slack Time                    0.914
     Clock Rise Edge                       0.000
     = Beginpoint Arrival Time             0.000
     +-------------------------------------------------------------------+
     | Instance |     Arc     |  Cell   | Delay | Arrival | Required |
     |          |             |         |       |  Time   |   Time   |
     |----------+-------------+---------+-------+---------+----------|
     |          | tclk ^      |         |       |   0.000 |    0.914 |
     | CG/BC1   | A ^ -> Y ^  | BUFX2   | 0.112 |   0.112 |    1.026 |
     | TC1      | A ^ -> Y ^  | BUFX2   | 0.155 |   0.267 |    1.181 |
     | CG1      | B ^ -> Y ^  | AND2X4  | 0.126 |   0.394 |    1.307 |
     | BLK/BC1  | A ^ -> Y ^  | BUFX2   | 0.130 |   0.523 |    1.437 |
     | BLK/BR2  | CK ^ -> Q v | DFF     | 0.265 |   0.789 |    1.702 |
     | BLK/BU5  | A v -> Y v  | BUFX2   | 0.136 |   0.924 |    1.838 |
     | BLK/BU6  | A v -> Y v  | BUFX2   | 0.132 |   1.056 |    1.970 |
     | TR2/BR1  | D v         | DFF     | 0.000 |   1.056 |    1.970 |
     +-------------------------------------------------------------------+
```

Example 32-20 `report_timing -point_to_point` Sample Reports

```
report_timing -unconstrained -point_to_point -to {buf12/Y buf13/Y}

Path 1: Endpoint: buf13/Y (v)
Beginpoint: clk      (v) (unconstrained input)
    Clock Rise Edge                     0.00
    + Input Delay                       0.00
    = Beginpoint Arrival Time           0.00
    +-------------------------------------------------+
    | Instance |     Arc     |  Cell | Delay | Arrival |
    |          |             |       |       |  Time   |
    |----------+-------------+-------+-------+---------|
    |          | clk v       |       |       |    0.00 |
    | buf11    | A v -> Y v  | BUFX1 | 10.00 |   10.00 |
    | buf12    | A v -> Y v  | BUFX1 |  0.16 |   10.16 |
    | buf13    | A v -> Y v  | BUFX1 |  0.15 |   10.30 |
    +-------------------------------------------------+
Path 2: Endpoint: buf12/Y (v)
Beginpoint: clk      (v) (unconstrained input)
    Clock Rise Edge                     0.00
    + Input Delay                       0.00
    = Beginpoint Arrival Time           0.00
    +-------------------------------------------------+
    | Instance |     Arc     |  Cell | Delay | Arrival |
    |          |             |       |       |  Time   |
    |----------+-------------+-------+-------+---------|
    |          | clk v       |       |       |    0.00 |
    | buf11    | A v -> Y v  | BUFX1 | 10.00 |   10.00 |
    | buf12    | A v -> Y v  | BUFX1 |  0.16 |   10.16 |
    +-------------------------------------------------+

report_timing -unconstrained -point_to_point -from clk -to {buf13/A buf1/Y out}
```

```
#when clocks are created

Path 1: Endpoint:   out (v) (unconstrained output)
Beginpoint: dff2/Q  (v) triggered by leading edge of 'CLK'
    Clock Rise Edge                     0.00
    = Beginpoint Arrival Time           0.00
    +-------------------------------------------------------+
    | Instance |     Arc     |  Cell   | Delay | Arrival |
    |          |             |         |       |  Time   |
    |----------+-------------+---------+-------+---------|
    |          | clk ^       |         |       |    0.00 |
    | buf11    | A ^ -> Y ^  | BUFX1   | 10.00 |   10.00 |
    | buf12    | A ^ -> Y ^  | BUFX1   |  0.13 |   10.13 |
    | buf13    | A ^ -> Y ^  | BUFX1   |  0.12 |   10.25 |
    | dff2     | CK ^ -> Q v | DFF     |  1.00 |   11.25 |
    | buf1     | A v -> Y v  | BUFX1   |  1.00 |   12.25 |
    | buf2     | A v -> Y v  | BUFX1   |  0.15 |   12.40 |
    |          | out v       |         |  0.00 |   12.40 |
    +-------------------------------------------------------+
Path 2: Endpoint: buf1/Y (v) (unconstrained output)
Beginpoint: dff2/Q  (v) triggered by leading edge of 'CLK'
    Clock Rise Edge                     0.00
    = Beginpoint Arrival Time           0.00
    +-------------------------------------------------------+
    | Instance |     Arc     |  Cell   | Delay | Arrival |
    |          |             |         |       |  Time   |
    |----------+-------------+---------+-------+---------|
    |          | clk ^       |         |       |    0.00 |
    | buf11    | A ^ -> Y ^  | BUFX1   | 10.00 |   10.00 |
    | buf12    | A ^ -> Y ^  | BUFX1   |  0.13 |   10.13 |
    | buf13    | A ^ -> Y ^  | BUFX1   |  0.12 |   10.25 |
    | dff2     | CK ^ -> Q v | DFF     |  1.00 |   11.25 |
    | buf1     | A v -> Y v  | BUFX1   |  1.00 |   12.25 |
    +-------------------------------------------------------+
Path 3: Endpoint: buf13/A (v) (unconstrained output)
Beginpoint: clk   (v) triggered by trailing edge of 'CLK'
    Clock Rise Edge                     2.00
    = Beginpoint Arrival Time           2.00
    +-------------------------------------------------------+
    | Instance |     Arc     |  Cell   | Delay | Arrival |
    |          |             |         |       |  Time   |
    |----------+-------------+---------+-------+---------|
    |          | clk v       |         |       |    2.00 |
    | buf11    | A v -> Y v  | BUFX1   | 10.00 |   12.00 |
    | buf12    | A v -> Y v  | BUFX1   |  0.16 |   12.16 |
    | buf13    | A v         | BUFX1   |  0.00 |   12.16 |
    +-------------------------------------------------------+


report_timing -unconstrained -point_to_point -from buf11/Y /
          -to {buf13/A buf1/Y out}
```

```
  #when -from is intermediate begin point


Path 1: Endpoint:   out (v) (unconstrained output)
Beginpoint: dff2/Q  (v) triggered by leading edge of 'CLK'
    +-------------------------------------------------------+
    | Instance |     Arc     |  Cell   | Delay | Arrival |
    |          |             |         |       |  Time   |
    |----------+-------------+---------+-------+---------|
    | buf11    | Y ^         |         |       |    0.00 |
    | buf12    | A ^ -> Y ^  | BUFX1   |  0.13 |    0.13 |
    | buf13    | A ^ -> Y ^  | BUFX1   |  0.12 |    0.25 |
    | dff2     | CK ^ -> Q v | DFF     |  1.00 |    1.25 |
    | buf1     | A v -> Y v  | BUFX1   |  1.00 |    2.25 |
    | buf2     | A v -> Y v  | BUFX1   |  0.15 |    2.40 |
    |          | out v       |         |  0.00 |    2.40 |
    +-------------------------------------------------------+
Path 2: Endpoint: buf1/Y (v) (unconstrained output)
Beginpoint: dff2/Q  (v) triggered by leading edge of 'CLK'
    +-------------------------------------------------------+
    | Instance |     Arc     |  Cell   | Delay | Arrival |
    |          |             |         |       |  Time   |
    |----------+-------------+---------+-------+---------|
    | buf11    | Y ^         |         |       |    0.00 |
    | buf12    | A ^ -> Y ^  | BUFX1   |  0.13 |    0.13 |
    | buf13    | A ^ -> Y ^  | BUFX1   |  0.12 |    0.25 |
    | dff2     | CK ^ -> Q v | DFF     |  1.00 |    1.25 |
    | buf1     | A v -> Y v  | BUFX1   |  1.00 |    2.25 |
    +-------------------------------------------------------+
Path 3: Endpoint:  buf13/A (v) (unconstrained output)
Beginpoint: buf11/Y   (v) triggered by trailing edge of 'CLK'
    +-------------------------------------------------------+
    | Instance |     Arc     |  Cell   | Delay | Arrival |
    |          |             |         |       |  Time   |
    |----------+-------------+---------+-------+---------|
    | buf11    | Y ^         |         |       |    0.00 |
    | buf12    | A v -> Y v  | BUFX1   |  0.16 |    0.16 |
    | buf13    | A v         | BUFX1   |  0.00 |    0.16 |
    +-------------------------------------------------------+
```

- The following command recalculates the specified critical paths using the LOCV factor and generates the following timing report:

```
report_timing -from 1_2/CK -path_type full_clock -retime locv

Path 1: MET Setup Check with Pin i_4/CK
Endpoint:   i_4/D (v) checked with leading edge of 'CLK'
Beginpoint: i_2/Q (v) triggered by leading edge of 'CLK'
Other End Arrival Time         0.281
- Setup                        0.291
+ Phase Shift                  2.000
```

```
+ Required Time                1.990
- Arrival Time                 0.851
+ Slack Time                   1.139
+ Slack Time (underated)       1.076
    Clock Rise Edge                        0.000
    + Beginpoint Arrival Time              0.000
    Timing Path:
```

Instance	Arc	Cell	Retime Delay	Arrival Time	Required Time
	clk1 ^			0.000	1.139
i_6		BUFX1	0.000	0.000	1.139
i_6	A ^ -> Y ^	BUFX1	0.115	0.115	1.254
i_7		BUFX1	0.000	0.115	1.254
i_7	A ^ -> Y ^	BUFX1	0.157	0.272	1.411
i_2		DFFRX2	0.000	0.272	1.411
i_2	CK ^ -> Q v	DFFRX2	0.446	0.718	1.856
i_3		BUFX1	0.000	0.718	1.856
i_3	A v -> Y v	BUFX1	0.133	0.851	1.990
i_4		DFFRX2	0.000	0.851	1.990

```
    Clock Rise Edge                        0.000
    = Beginpoint Arrival Time              0.000
    Other End Path:
```

Instance	Arc	Cell	Retime Delay	Arrival Time	Required Time
	clk1 ^			0.000	-1.139
i_6		BUFX1	0.000	0.000	-1.139
i_6	A ^ -> Y ^	BUFX1	0.057	0.057	-1.082
i_7		BUFX1	0.000	0.057	-1.082
i_7	A ^ -> Y ^	BUFX1	0.072	0.129	-1.010
i_8		BUFX1	0.000	0.129	-1.009
i_8	A ^ -> Y ^	BUFX1	0.053	0.182	-0.957
i_9		BUFX1	0.000	0.182	-0.957
i_9	A ^ -> Y ^	BUFX1	0.047	0.229	-0.909
i_10		BUFX1	0.000	0.229	-0.909
i_10	A ^ -> Y ^	BUFX1	0.051	0.280	-0.858
i_4		DFFRX2	0.000	0.281	-0.858

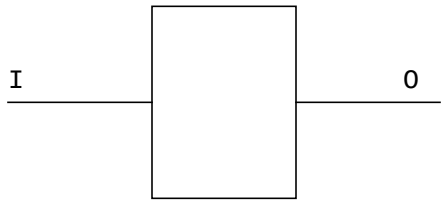
- The following command recalculates the specified critical paths using the LOCV factor and generates an LOCV derating summary table for the launch and capture paths, in addition to the timing report:

```
report_timing -from i_2/CK -path_type full_clock -retime locv /
       -derate_summary -format {instance cell delay arrival locv_derate}

Path 1: MET Setup Check with Pin i_4/CK
Endpoint:   i_4/D (v) checked with leading edge of 'CLK'
Beginpoint: i_2/Q (v) triggered by leading edge of 'CLK'
Other End Arrival Time         0.281
- Setup                        0.291
+ Phase Shift                  2.000
+ Required Time                1.990
- Arrival Time                 0.851
+ Slack Time                   1.139
+ Slack Time (underated)       1.076
    Clock Rise Edge                      0.000
    + Beginpoint Arrival Time            0.000
    Timing Path:

   +-------------------------------------------------+
   | Instance | Cell      | Delay  | Arrival | LOCV   |
   |          |           |        |  Time   | Derate |
   |----------+-----------+--------+---------+--------|
   |          |           |        |  0.000  |        |
   | i_6      | BUFX1     | 0.000  |  0.000  |  1.000 |
   | i_6      | BUFX1     | 0.115  |  0.115  |  1.000 |
   | i_7      | BUFX1     | 0.000  |  0.115  |  1.000 |
   | i_7      | BUFX1     | 0.157  |  0.272  |  1.000 |
   | i_2      | DFFRX2    | 0.000  |  0.272  |  0.873 |
   | i_2      | DFFRX2    | 0.446  |  0.718  |  0.873 |
   | i_3      | BUFX1     | 0.000  |  0.718  |  0.873 |
   | i_3      | BUFX1     | 0.133  |  0.851  |  0.873 |
   | i_4      | DFFRX2    | 0.000  |  0.851  |  0.873 |
   +-------------------------------------------------+

    Clock Rise Edge                     0.000
    = Beginpoint Arrival Time           0.000
```

```
 Other End Path:

+-------------------------------------------------+
| Instance | Cell      | Delay | Arrival | LOCV   |
|          |           |       |  Time   | Derate |
|----------+-----------+-------+---------+--------|
|          |           |       |  0.000  |        |
| i_6      | BUFX1     | 0.000 |  0.000  |  1.000 |
| i_6      | BUFX1     | 0.057 |  0.057  |  1.000 |
| i_7      | BUFX1     | 0.000 |  0.057  |  1.000 |
| i_7      | BUFX1     | 0.072 |  0.129  |  1.000 |
| i_8      | BUFX1     | 0.000 |  0.129  |  0.877 |
| i_8      | BUFX1     | 0.053 |  0.182  |  0.877 |
| i_9      | BUFX1     | 0.000 |  0.182  |  0.877 |
| i_9      | BUFX1     | 0.047 |  0.229  |  0.877 |
| i_10     | BUFX1     | 0.000 |  0.229  |  0.877 |
| i_10     | BUFX1     | 0.051 |  0.280  |  0.877 |
| i_4      | DFFRX2    | 0.000 |  0.281  |  0.877 |
+-------------------------------------------------+

DUAL LOCV Derate Summary
Clock Branch Point    i_7/Y
Distance   36.9000


+--------------------------+
| Launch Path Derate Table |
|--------------------------|
| Element | Stage | Derate |
| Type    | Count |        |
|---------+-------+--------|
| Cell    |   2   |  0.873 |
| Net     |   3   |  0.873 |
+--------------------------+

+--------------------------+
|    Capture Path Derate   |
|          Table           |
|--------------------------|
| Element | Stage | Derate |
| Type    | Count |        |
|---------+-------+--------|
| Cell    |   3   |  0.877 |
| Net     |   4   |  0.877 |
+--------------------------+
```

The following examples refer to Figure 32-1 .

Figure 32-1 Conditions for Unconstrained and Constrained Paths



- Figure 32-1 shows the path from `I` to `O` is reported as an unconstrained path under the following conditions (constraints on `I` and `O`):
  - No constraints on either `I` or `O`
  - `set_input_delay -clock ck1 1.0 I`
    No `set_output_delay` constraint on `O`
  - `set_input_delay 1.0 I`
    `set_output_delay 1.1 O`
- Shows the path from `I` to `O` is reported as a constrained path under the following conditions (constraints on `I` and `O`):
  - `set_input_delay -clock ck1 1.0 I`
    `set_output_delay -clock ck1 1.0 O`
  - `set_input_delay -clock ck1 1.0 I`
    `set_output_delay -clock ck2 1.0 O`
  - `set_output_delay -clock ck1 1.0 O`
    No `set_input_delay` constraint on I. It uses default delay of zero.

  Default: `-late`

Related Information

`report_net`

`report_ports`

reportAnalysisMode

```
reportAnalysisMode
```

Reports the timing analysis mode settings for the current Encounter session.

Parameters

None

reportClockDomains

`reportClockDomains -outfile` *`fileName`*

Reports the clock domain settings after you run timing analysis.

Parameters

`-outfile` *`fileName`*	Specifies the report output file. If you do not specify this parameter, the report is displayed on the Encounter console.

Command Order

Use this command after running timing analysis.

Example

The following command reports the clock domain information to the file `TOPCHIP.clkRpt`:

```
reportClockDomains -outfile TOPCHIP.clkRpt
```

reportTimingDerate

```
reportTimingDerate
     [-delay_corner delayCornerName]
```

Reports delay scaling or derating factors for early and late clock and data paths in the current design.

Parameters

`-delay_corner` *`delayCornerName`*

Reports derating factors for the specified delay calculation corner only.

Default: Reports derating factors for all delay calculation corners.

Related Information

`set timing derate`

reportTimingLib

```
reportTimingLib
     -outfile fileName
     [-byLib]
     [-all | -pin | -threshold | -sibling]
     [cellName]
```

Reports the contents of the timing library.

Parameters

`-all | -pin | -threshold | -sibling`

Specifies the parameter for which the software should report the timing data.

`-all` prints all the pin properties, siblings of a cell, the libraries where they are defined, and the voltage.

`-pin` reports the pin properties, such as the pin name, direction, capacitance, max load, max fanout, and fanout load.

`-threshold` reports the input thresholds, output thresholds, and slew thresholds of a cell.

`-sibling` reports sibling cells, the libraries where they are defined, and the voltage.
Default: `-all`

`-byLib`

Organizes the generated report by libraries. If you do not specify this parameter, the `reportTimingLib` command prints out cells with the same name together. Use this parameter to support multi-supply multi-voltage feature of the software, where you can define same cell names in more than one library with different voltage values and timing data.

`-outfile fileName`

Specifies the name of the file to which the results are written.

`cellName`

Specifies the individual cell name to be reported.

Command Order

Use this command after importing the design when the timing library is also loaded.

Example

The following command reports the cell information in the timing library, and writes the results to the file `xyz.dump.lib`:

```
reportTimingLib -outfile xyz.dump.lib
```

reset_sdf_assertions

```
reset_sdf_assertions
```

Resets the sdf delays to the values defined in the library. When you perform timing analysis using the timing engine, the software uses sdf values that you define using the `read_sdf` command earlier in the flow. You use the `reset_sdf_assertions` commands to discard delay values specified using the sdf files, and use the delays defined in the library.

Note: You do not need to read in the sdf delays every time you perform delay calculation. The software uses the sdf files specified earlier in the flow.

Parameter

None

reset_timing_derate

```
reset_timing_derate
     [-delay_corner delayCornerName]
     [object_list]
```

Resets derate factors specified on a design or a list of instances (cells, net, or library cells). All the derating factors that are set globally or on specific instances are reset to a default value of `1.0`.

Derating factors for early or late paths in the design are set using the set_timing_derate command.

Parameters

`-delay_corner` *`delayCornerName`*

Specifies the delay corner on which reset derating is to be applied.

Note: In multi-mode multi-corner analysis mode, you are required to specify the `-delay_corner` parameter.

`object_list`

Specifies the list of cells, nets, or library cells to reset.

If the *`object_list`* contains hierarchical cells then all cells within the hierarchical cells and across lower hierarchies have reset derate factors.

Examples

- The following example resets all the object-specific and global derating factors:

  ```
  reset_timing_derate *
  ```

- The following example resets only global derating factors:

  ```
  reset_timing_derate [current_design]
  ```

- The following example resets derating factor values for the specified delay corner:

  ```
  reset_timing_derate -delay_corner delayCornerName
  ```

- The following example resets the global derating values for the specified delay corner:

  ```
  reset_timing_derate -delay_corner delayCornerName [current_design]
  ```

- The following example resets timing derate on instance u1:

  ```
  reset_timing_derate [get_cells u1]
  ```

select_locv_table

```
select_locv_table
     -setup [min | max]
     -hold [min | max]
     [-view view_name]
```

Specifies the type of LOCV table to use for setup or hold analysis. When you specify the `select_locv_table` command, the software looks in the LOCV library file for an LOCV table that matches your specifications, and derives the derating factors from that table.

Each LOCV table in the library is defined with a combination of the following settings:

- `MAX` (worst) and `MIN` (best)
- `Setup` and `Hold`
- `Early` and `Late`
- `Rise` and `Fall`

You can use the `select_locv_table` command to specify whether to use a `MAX` or `MIN` LOCV table for setup or hold analysis. The software automatically chooses the `Setup`/`Hold`, `Early`/`Late`, and `Rise`/`Fall` combinations that should be used.

Parameters

`-hold [min | max]`	Specifies whether to use the min or max condition for hold analysis. *Default:* `min`
`-setup [min | max]`	Specifies whether to use the min or max condition for setup analysis. *Default:* `max`
`-view view_name`	Selects the LOCV table to use for the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

set_analysis_view

```
set_analysis_view
     -setup {defaultSetupView setupView2 ...}
     -hold {defaultHoldView holdView2 ...}
```

Defines the analysis views to use for setup and hold analysis and optimization. These “active” views represent the different design variations that will be analyzed. Active views can be changed throughout the flow to utilize different subsets of views. Encounter applications can handle the views concurrently or sequentially, depending on their individual capabilities. Libraries and data are loaded into the system, as required to support the selected set of active views.

You must define at least one setup and one hold analysis view.

The order in which you specify views with the `-setup` and `-hold` parameters is important. By default, the first views defined in the `-setup` and `-hold` lists are the default views. Certain Encounter applications that do not support multi-mode multi-corner can only process the data defined for a single view. These applications use the information defined for the default view.

Note: Use the `set_default_view` command to change the default analysis views. Using the `set_default_view` command does not affect software performance because it only uses views that are already active in the design. If you use the `set_analysis_view` command to change the default views, the existing timing, delay calculation, and RC data is reset.

Use this command after creating multiple analysis views (`create_analysis_view`).

Parameters

`-hold {`*`defaultHoldView holdView2 …`*`}`

Sets the active views for hold analysis.

`-setup {`*`defaultSetupView setupView2 …`*`}`

Sets the active views for setup analysis.

Examples

- The following command sets `missionWCCOM` and `mission2WCCOM` as the active views for setup analysis, and `missionBCCOM` and `testBCCOM` as the active views for hold analysis:

```
set_analysis_view -setup {missionWCCOM mission2WCCOM}
      -hold {missionBCCOM testBCCOM}
```

Related Topics

- Performing Multi-Mode Multi-Corner Timing Analysis and Optimization chapter in the *Encounter User Guide*
  - "Configuring the Setup for Multi-Mode Multi-Corner Analysis"
  - "Setting Active Analysis Views"
  - "Guidelines For Setting Active Analysis Views"
  - "Changing the Default Active Analysis View"
- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- mmmc.tcl in the *Encounter Hierarchical Implementation Flow Guide*

set_default_view

```
set_default_view
     {-setup newDefaultSetupView | -hold newDefaultHoldView}
```

Temporarily changes the default active analysis view to a different active view.

Some Encounter applications can function only on a single analysis view at a time. By default, these single-view applications use the default analysis view. If an application or flow step does not provide a native option for specifying which view to use, the `set_default_view` command can be used to temporarily change the default view to a different active view that is better suited to that flow step.

Note: Using the `set_default_view` command does not affect software performance because it only uses views that are already active in the design. If you use the `set_analysis_view` command to change the default views, the existing timing, delay calculation, and RC data is reset.

Use this command after defining active views for the design (`set_analysis_view`).

Parameters

`-hold` *`newDefaultHoldView`*

Changes the default active hold analysis view to the specified analysis view.

`-setup` *`newDefaultSetupView`*

Changes the default setup analysis view to the specified analysis view.

Examples

- The following command sets `missionWCCOM`, `mission2WCCOM`, and `missionSlow` as the active views for setup analysis, and `missionBCCOM` and `testBCCOM` as the active views for hold analysis. The default active setup view is `missionWCCOM`, and the default active hold view is `missionBCCOM`.

  ```
  set_analysis_view -setup {missionWCCOM mission2WCCOM missionSlow}
        -hold {missionBCCOM testBCCOM}
  ```

- The following command changes the default active setup view to `missionSlow`:

  ```
  set_default_view -setup missionSlow
  ```

- The following command changes the default active hold view to `testBCCOM`:

```
set_default_view -hold testBCCOM
```

Related Topics

- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- mmmc.tcl in the *Encounter Hierarchical Implementation Flow Guide*

set_guard_band_derate

```
set_guard_band_derate
     [-early guard_band_factor]
     [-late guard_band_factor]
     [-view view_name]
```

Adjusts the LOCV derating values extracted from the LOCV library by the specified factor.

Use guard band derating to add pessimism to account for differences in such things as extraction and delay calculation.

Parameters

`-early` *`guard_band_factor`*

Specifies the percentage by which to adjust the LOCV derating factor for early paths.

`-late` *`guard_band_factor`*

Specifies the percentage by which to adjust the LOCV derating factor for late paths.

`-view` *`view_name`* Adjusts the LOCV derating factor for the specified analysis view only.

You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Examples

- Assuming that the LOCV derating factors are 1.0 for early paths and 2.0 for late paths, the following command adjusts the LOCV derating factors to 0.9 for early paths (10 percent earlier) and 2.4 for late paths (20 percent later):

  ```
  set_guard_band_derate -early 0.1 -late 0.2
  ```

set_interactive_constraint_modes

`set_interactive_constraint_modes {`*`list_of_constraint_modes`*`}`

Puts the software into interactive constraint entry mode for the specified multi-mode multi-corner constraint mode objects. Any timing constraints that you specify after this command take effect immediately on all active analysis views that are associated with the specified constraint modes. By default, no constraint modes are considered active.

The software stays in interactive mode until you exit by specifying the `set_interactive_constraint_modes` command with an empty list:

```
set_interactive_constraint_modes { }
```

All new assertions are saved in the SDC files of the specified constraint modes when you save the design (`saveDesign`).

The `all_constraint_modes` command can be used to generate a list of constraint modes as the argument for this command.

Use the `get_interactive_constraint_modes` command to return a list of the constraint mode objects in interactive constraint entry mode.

Note: Interactive constraint mode only works when the software is in multi-mode multi-corner timing analysis mode. In min/max analysis mode, constraints are always accepted interactively.

Parameters

`{`*`list_of_constraint_modes`*`}`

Specifies the constraint mode objects to update with the new assertions.

Examples

- The following commands put the software into interactive constraint entry mode, and apply the `set_propagated_clock` assertion on all views in the current session that are associated with the constraint mode `func1`:

  ```
  set_interactive_constraint_modes func1
  set_propagated_clock [all_clocks]
  ```

  When you specify `saveDesign`, the updated constraints file for the `func1` constraint mode will include the `set_propagated_clock`.

- The following commands put the software into interactive constraint entry mode, and apply the `set_propagated_clock` assertion on all active analysis views in the current session.

  ```
  set_interactive_constraint_modes [all_constraint_modes -active]
  set_propagated_clock [all_clocks]
  ```

  When you specify `saveDesign`, the `set_propagated_clock` assertion will be saved persistently as part of a new `.sdc` file.

Related Topics

- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- mmmc.tcl in the *Encounter Hierarchical Implementation Flow Guide*

set_io_thresholds

```
set_io_thresholds
     [-slew_lower_threshold_pct_rise pctValue]
     [-slew_lower_threshold_pct_fall pctValue]
     [-slew_upper_threshold_pct_rise pctValue]
     [-slew_upper_threshold_pct_fall pctValue]
     [-input_threshold_pct_rise pctValue]
     [-input_threshold_pct_fall pctValue]
     [-output_threshold_pct_rise pctValue]
     [-output_threshold_pct_fall pctValue]
```

Sets the threshold trip-points used for delay calculation to and from the primary I/Os. The `set_io_thresholds` command does not affect the interpretation of threshold settings in the Liberty timing libraries; the effect is strictly limited to the I/O ports.

If you do not specify I/O thresholds with the `set_io_thresholds` command, delay calculation assumes that the I/O thresholds are equivalent to those of the logic gates to which they are connected.

The `set_io_thresholds` command supports separate rise and fall slew thresholds as normally found in Liberty timing libraries.

Note: Some third-party applications derive the I/O threshold values from the first Liberty timing library in the library search path. The software currently does not support this methodology. Explicit setting of I/O thresholds with the `set_io_thresholds` command should be used to complete a compatible analysis environment.

Use this command immediately after loading the configuration file or restoring the design.

Parameters

`-input_threshold_pct_fall` *`pctValue`*

Specifies the measurement point for a falling signal that will be used for calculating the interconnect delay from a primary input to internal logic.

Value: `0 - 100`

Default: `50`

`-input_threshold_pct_rise` *`pctValue`*

Specifies the measurement point for a rising signal that will be used for calculating the interconnect delay from a primary input to internal logic.

Value: `0 - 100`

Default: `50`

`-output_threshold_pct_fall` *`pctValue`*

Specifies the measurement point for a falling signal that will be used for calculating the interconnect delay from internal logic to a primary output.

Value: `0 - 100`

Default: `50`

`-output_threshold_pct_rise` *`pctValue`*

Specifies the measurement point for a rising signal that will be used for calculating the interconnect delay from internal logic to a primary output.

Value: `0 - 100`

Default: `50`

`-slew_lower_threshold_pct_fall` *`pctValue`*

Specifies the lower trip-point for a falling signal that will be used to measure transition times.

Value: `0 - 100`

Default: `20`

`-slew_lower_threshold_pct_rise` *`pctValue`*

Specifies the lower trip-point for a rising signal that will be used to measure transition times.

Value: `0 - 100`

Default: `20`

`-slew_upper_threshold_pct_fall` *`pctValue`*

Specifies the upper trip-point for a falling signal that will be used to measure transition times.

Value: `0 - 100`

Default: `80`

`-slew_upper_threshold_pct_rise` *`pctValue`*

Specifies the upper trip-point for a rising signal that will be used to measure transition times.

Value: `0 - 100`

Default: `80`

Examples

- The following command specifies trip-points for measuring delay and transition times:

```
set_io_thresholds -slew_lower_threshold_pct_rise 10
                  -slew_upper_threshold_pct_fall 80
                  -slew_upper_threshold_pct_rise 90
                  -slew_lower_threshold_pct_fall 20
                  -input_threshold_pct_rise 40
                  -input_threshold_pct_fall 70
                  -output_threshold_pct_rise 60
                  -output_threshold_pct_fall 30
```

setOpCond

```
setOpCond
     opCond
     [-library libName]
     [-min minOpCond [-minLibrary minLib]]
     [-max maxOpCond [-maxLibrary maxLib]]
     [-powerDomain domainName]
     [-forceLibrary libName]
```

Sets the operating temperature, process, or voltage conditions for the design. The operating conditions are contained in the timing library, which are normally read in during design import.

The Setting Operating Conditions section in the *Encounter User Guide* provides more information on how to use this command to set the operating conditions for analysis.

Important

You cannot use this command when the software is in multi-mode multi-corner timing analysis mode. Instead, use the `create_delay_corner` and `update_delay_corner` commands to set operating conditions.

The `setOpCond` command can use a virtual operating condition that was created with the multi-mode multi-corner `create_op_cond` command.

Parameters

`-forceLibrary` *libName*

Specifies the library in which the *opCond* will be set. Do not use this parameter with the `-min`, `-max`, or `-powerDomain` parameters. Use this parameter to specify different operating conditions for different libraries in a single session.

Note: This parameter is provided only for backward compatibility with previous versions of Encounter. Cadence does not recommend the use of this parameter.

`-library` *`libName`*

Specifies the library in which the operating condition (*`opcond`*) is defined. You can define a single operating condition from a library, and the software uses the PVT characteristics associated with that library for every library in the design. When you use this parameter in multiple `setOpCond` commands, the software uses the value specified in the last command. If the `-library` parameter is not specified, the *`opCond`* in the first library read is used.

For example, consider the following commands:

```
setOpCond WCCOM -library wcTestLib
setOpCond BCCOM -library bcTestLib
setOpCond WCCOM -library wcTestLib1
setOpCond BCCOM -library bctestLib1
setOpCond -library wcTestLib2
setOpCond -library bcTestLib2
```

In this case, the software uses the last value defined (`bcTestLib2`). Therefore, the operating conditions in the `bcTestLib2` library are used for both setup and hold analysis.

`-max` *`maxOpCond`*

Specifies the operating condition normally used for setup analysis.

`-maxLibrary` *`maxLib`*

Specifies the library in which the *`maxOpCond`* is defined. If the `-maxLibrary` parameter is not specified and `-max` is used, the *`maxOpCond`* from the first library listed in the maximum libraries is used.

`-min` *`minOpCond`*

Specifies the operating condition normally used for hold analysis.

`-minLibrary` *`minLib`*

Specifies the library in which the *`minOpCond`* is defined. If the `-minLibrary` parameter is not specified and `-min` is used, the *`minOpCond`* from the first library listed in the minimum libraries is used.

`opCond`

Specifies the operating condition used for setup and hold analysis.

You can specify a virtual operating condition created with the multi-mode multi-corner `create op cond` command.

`-powerDomain` *`domainName`*

Specifies that the *`opCond`* is used for the power domain.

Command Order

Use this command after importing the design and reading the libraries.

Examples

- The following command sets the operating condition of the design to `WCCOM` defined in the `slowLib` timing library. The worst case (WCCOM) operation condition will be used for both setup and hold analysis:

  ```
  setOpCond WCCOM -library slowLib
  ```

- The following Tcl command sets the operating condition to `WCCOM` defined in the `slowlib` library for setup analysis and sets the operating condition to `BCCOM` defined in the `fastlib` library for hold analysis:

  ```
  setOpCond -max WCCOM -maxLibrary slowlib -min BCCOM -minLibrary fastlib
  ```

- The following example shows that if you do not set the `setOpCond` command properly, you will get inconsistent results.

  - The following example specifies the `slow` operating condition defined in the `slow` library for setup analysis and the `fast` operating condition defined in the `fast` library for hold analysis:

    ```
    setOpCond -maxLibrary slow -max slow -minLibrary fast -min fast
    ```

  - The following example sets the timing libraries to associate the fast library (Min Libraries) to setup and the slow library (Max Libraries) to hold:

    ```
    setTimingLibrary -max Min -min Max
    ```

    However, this means that setup analysis uses the `fast` library (defined in the Min Libraries list) with a `slow` operating condition defined in the `slow` library, and hold analyses uses the `slow` library (defined in the Max Libraries list) but with a `fast` operating condition defined in the `fast` library.

  - To get a consistent result, reset the `setOpCond` command to:

    ```
    setOpCond -maxLibrary fast -max fast -minLibrary slow -min slow
    ```

    Now setup analysis uses the `fast` library (defined in the Min Libraries list) with a `fast` operating condition defined in the `fast` library, and hold analyses uses the `slow` library (defined in the Max Libraries list) but with a `slow` operating condition defined in the `slow` library.

set_table_style

```
set_table_style
     {  -name table_name
          [-reverse_rows]
          [-major_sort integer]
          [-minor_sort integer]
          [-max_widths list_of_integers]
          [-min_widths list_of_integers]
          [-indent integer]
     | -frame | -no_frame | -no_frame_fix_width [-nosplit]
     }
```

Disables printing of specific columns and allows for specifying the minimum and maximum size of each column in a report. It allows for reversing the data, for controlling the left indent, and for controlling the sorting of the columns.

Note: Do *not* include a blank space between commas "`,`" when specifying widths. If you are specifying a value for the width(s), make sure there is *no* leading or trailing blank space.

The `-frame`, `-no_frame`, and `-no_frame_fix_width` parameters work in a global manner. If you specify one of these parameters, it will affect all reports that can be formatted with this command. You cannot specify them with the -name parameter, to change the format of a specific report.

Parameters

`-frame`	Specifies that the table columns have a fixed frame or border, based on the minimum and maximum width settings. Any data that is longer than the maximum width for its column is wrapped within the same column. This is the default format for tables.
`-indent` *`integer`*	Specifies the number of spaces to leave on the left for that particular table.
`-major_sort` *`integer`*	Specifies the column to be used in the major sort. If the column number is negative, it reverses the sort.

`-max_widths` *`list_of_integers`*

Specifies the maximum width for each column in the table. Separate each specified width value with a comma (`,`). For example, if you specify:

```
-max_widths {10,20,30}
```

The first column will have a maximum width of 10, the second column will have a maximum width of 20, and the third column will have a maximum width of 30. The fourth and fifth columns will remain the default width spacing.

Commas are also specified as column “placeholders”, when you want to change the spacing of a column without changing the spacing of the preceding column. For example, if you specify:

```
-max_widths {,,20,30}
```

The third column will have a maximum width of 20, the fourth column will have a maximum width of 30, and the first, second, and fifth columns will remain the default width spacing.

If you specify:

```
-max_widths {20,,30}
```

The first column will have a maximum width of 20, the third column will have a maximum width of 30, and the second, fourth and fifth columns will remain the default width spacing.

Each column resizes itself to be between the range of the maximum and minimum widths specified for that column, based on the data in that column. Overruns are wrapped within the same column, unless otherwise specified.

Default: `{50,50,50,50,50}`

`-min_widths` *`list_of_integers`*

Specifies the minimum width for each column in the table. Separate each specified width value with a comma (`,`). For example, if you specify:

```
-min_widths {7,10,15}
```

The first column will have a minimum width of 7, the second column will have a minimum width of 10, and the third column will have a minimum width of 15. The fourth and fifth columns will remain the default width spacing.

Commas are also specified as column “placeholders”, when you want to change the spacing of a column without changing the spacing of the preceding column. For example, if you specify:

```
-min_width {,,10,15}
```

The third column will have a minimum width of 10, the fourth column will have a minimum width of 15, and the first, second, and fifth columns will remain the default width spacing.

If you specify:

```
-min_width {7,,10}
```

The first column will have a minimum width of 7, the third column will have a width of 10, and the second, fourth and fifth columns will remain the default width spacing.

Each column resizes itself to be between the range of the maximum and minimum widths specified for that column, based on the data in that column. Overruns are wrapped within the same column, unless otherwise specified.

Default: `{5,5,5,5,5}`

`-minor_sort` *`integer`*

Specifies the column to be used in the minor sort. It applies to *all* rows that have the same values in the `major_sort` column. If the column number is negative, the software reverses the sort.

`-name` *`table_name`* Changes the formatting for the specified table. See Table 32-9 on page 1797 for a list of table names you can use, and the commands with which they work.

`-no_frame`	Removes intra-column wrapping: a column that contains data that is longer than the column width will take as much space in the table as needed (the data does not wrap within the column). If a column is overwritten by the data from the preceding column, its data will begin on a new line at the correct column location. Each column resizes itself to be between the range of the maximum and minimum widths specified for that column, based on the data in that column. *Default:* `-frame`
`-no_frame_fix_width`	Specifies that the table columns have a fixed minimum width, but no maximum width (the data does not wrap). Column data that fits within the column's minimum width will left-justify and pad its location to fill the specified column space. Column data that is longer than the column's minimum width will take as much space in the table as needed. If a column is overwritten by the data from the preceding column, its data will begin on a new line at the correct column location. *Default:* `-frame`
`-nosplit`	Specifies that each column follows the preceding column without splitting to a new line, if the data in one column overruns the next column. The wrapping behavior for data is controlled by the size of the display window. This format is useful for machine parsing the report. **Note:** You must specify `-no_frame_fix_width` in order to use this parameter.
`-reverse_rows`	Reverses the data rows. This is useful, for example, if you want to trace back from an end point and the default report traces forward from the end point.

Table 32-9 List of Table Names

Table Name	Formats Reports Generated by
`report_cell_instance_delays_arcs`	`report_cell_instance_timing`
`report_cell_instance_delays_checks`	`report_cell_instance_timing`

Table Name	Formats Reports Generated by
report_cell_instance_delays_pins	report_cell_instance_timing
report_inactive_arcs	report_inactive_arcs
report_net	report_net
report_net_header	report_net
report_net_sink_table	report_net
report_net_source_table	report_net
report_net_summary_table	report_net
report_path_exceptions	report_path_exceptions
report_port	report_ports
report_timing	report_timing
	report_constraint
report_timing_summary	report_timing

Examples

- The following command resizes the first column to a maximum range of 70 and specifies a default width of 50 for the second column. The default for the maximum width is 50 and the default for the minimum width is 5:

  ```
  set_table_style -name report_timing -max_widths {70,}
  ```

  Note: Do *not* include a blank space between commas "**,**" when specifying widths. If you are specifying a value for the width(s), make sure there is *no* leading or trailing blank space.

- The following command specifies a default width of 50 for the first and second column:

  ```
  set_table_style -name report_timing -max_widths {,}
  ```

- The following sample timing report shows the table style generated with the `-frame` parameter (`set_table_style -frame`):

```
+-------------------------------------------------------------------------+
|      Instance          |     Arc     |  Cell   | Delay | Arrival | Required  |
|                        |             |         |       |  Time   |   Time    |
|------------------------+-------------+---------+-------+---------+-----------|
|                        | in2 v       |         |       |   1.000 |     4.024 |
| buf133                 | A v -> Y v  | BUFX1   | 0.158 |   1.158 |     4.182 |
| bufbufbufbufbuf5bufbu  | A v -> Y v  | BUFX1   | 0.172 |   1.330 |     4.354 |
| f5buf5buf5buf5         |             |         |       |         |           |
| buf0                   | A v -> Y v  | BUFX1   | 0.184 |   1.513 |     4.538 |
| lat2                   | D v         | LATN1   | 0.000 |   1.513 |     4.538 |
+-------------------------------------------------------------------------+
```

- The following sample timing report shows the table style generated with the `-no_frame_fix_width` parameter (`set_table_style -no_frame_fix_width`):

```
-------------------------------------------------------------------------
Instance              Arc          Cell     Delay    Arrival    Required
                                                       Time       Time
-------------------------------------------------------------------------
                      in2 v                             1.000     4.024
buf133                A v -> Y v   BUFX1    0.158     1.158     4.182
bufbufbufbufbuf5bufbu
f5buf5buf5buf5        A v -> Y v   BUFX1    0.172     1.330     4.354
buf0                  A v -> Y v   BUFX1    0.184     1.513     4.538
lat2                  D v          LATN1    0.000     1.513     4.538
-------------------------------------------------------------------------
```

- The following sample timing report shows the table style generated with the `-no_frame_fix_width` and `-nosplit` parameters (`set_table_style -no_frame_fix_width -nosplit`):

```
  -------------------------------------------------------------------
  Instance          Arc          Cell    Delay    Arrival    Required
                                                   Time       Time
  -------------------------------------------------------------------
                    in2 v                             1.000     4.024
  buf133            A v -> Y v   BUFX1   0.158     1.158     4.182
  bufbufbufbufbuf5bufbuf5buf5buf5buf5buf5buf5buf5buf5buf5       A v -> Yv
BUFX1    0.172     1.330     4.354
  buf0              A v -> Y v   BUFX1   0.184     1.513     4.538
  lat2              D v          LATN1   0.000     1.513     4.538
  -------------------------------------------------------------------
```

Related Information

report_cell_instance_timing

report_inactive_arcs

report_net

report_path_exceptions

report_ports

report_timing

set_timing_derate

set_timing_derate
```
     [-early]
     [-late]
     [-min | -max | -delay_corner delayCornerName]
     [-clock]
     [-data]
     [-net_delay]
     [-cell_delay]
     [-retain_delay]
     [-cell_check]
     [-dynamic]
     [-static]
     derate_value
     [object_list]
```

Sets delay scaling or derating factors for early and late paths in the current design. You use this command to perform scaled on-chip variation around best-case and worst-case corners. The timing scaling factors affect the delay values generated in the timing reports. You can set scaling factors for data paths, clock paths, minimum and maximum operating conditions.

If you do not specify the `-clock` or `-data` parameter, the software applies the scaling factors to both clock and data paths. If you specify only one of the two parameters, `-data` or `-clock`, the software uses the last defined value for the other.

Similarly, if you do not specify the `-min` or `-max` parameter, the software applies the scaling factor to both minimum and maximum operating conditions. You should only use the `-min` or `-max` parameter for best-case worst-case analysis. If you specify them in on-chip variation (OCV) mode, the software will use the `-min -early` derating factor for early path scaling, and the `-max -late` derating factor for late path adjustments.

Note: The `set_timing_derate` command can only be used on the command line. You cannot import it as part of an SDC file.

Parameters

`-cell_check`	Specifies scaling on timing checks. Use the `-cell_check` option with the `-early` option to specify scaling on hold arcs and use with the `-late` option to specify scaling on setup arcs.
`-cell_delay`	Specifies scaling on cell delays. If neither the `-cell_delay` nor the `-net_delay` option is specified, the specified scaling factor applies to both types of delays.

`-clock` Applies the derating factors to clock paths only.

`-data` Applies the derating factors to data paths only.

`-delay_corner` *`delayCornerName`*

Applies the derating factor to the specified delay calculation corner.

`derate_value` Specifies the derating value to be used.

`-dynamic` Applies the specified derating factor to the delta portion of net delays only.

The `-net_delay` parameter must be specified with this parameter.

`-early` Specifies that the derating factor applies to early paths. Early paths are launching clock during hold analysis, receiving clock during setup analysis and datapath during hold analysis. For definition of early and late paths, see "Definition of Early Late Paths" in *Encounter User Guide*.

For backward compatibility with the Encounter `setTimingDerate` command, the `set_timing_derate` command supports the specification of an early and late derating factor using the same command. You can specify either:

```
set_timing_derate -early -clock derateValue
set_timing_derate -late -clock derateValue
```

or

```
set_timing_derate -early earlyDerateValue
    -late lateDerateValue -clock
```

Cadence recommends using the former approach, because it provides better consistency with the SDC specification, and therefore, with other timing analysis utilities.

`-late`	Specifies that the derating factor applies to late paths. Late paths are receiving clock during hold analysis, launching clock during setup analysis and datapath during setup analysis. For backward compatibility with the Encounter `setTimingDerate` command, the `set_timing_derate` command supports the specification of an early and late derating factor using the same command. You can specify either: `set_timing_derate -early -clock` *`derateValue`* `set_timing_derate -late -clock` *`derateValue`* or `set_timing_derate -early` *`earlyDerateValue`* `-late` *`lateDerateValue`* `-clock` Cadence recommends using the former approach, because it provides better consistency with the SDC specification, and therefore, with other timing analysis utilities.
`-max`	Applies the derating factor to the maximum operating condition during best-case worst-case analysis. This parameter is ignored during on-chip variation analysis. **Note:** You cannot use the `-max` parameter when the software is in multi-mode multi-corner (MMMC) mode. You must use the `-delay_corner` parameter instead.
`-min`	Applies the derating factor to the minimum operating condition during best-case worst-case analysis. This parameter is ignored during on-chip variation analysis. **Note:** You cannot use the `-min` parameter when the software is in multi-mode multi-corner (MMMC) mode. You must use the `-delay_corner` parameter instead.
`-net_delay`	Specifies scaling on net delays.
`object_list`	Specifies the list of instances, library cells, or nets to which the derating will be applied.
`-retain_delay`	Specifies scaling on retain constructs. If you specify both `-retain_delay` and `-cell_delay` parameters, the software uses the retain_delay value for retain construct scaling. If you do not specify the `-retain_delay` parameter, retain constructs are scaled in the same manner as cell delays.

`-static`	Applies the specified derating factor to the non-delta portion of net delays only. The `-net_delay` parameter must be specified with this parameter.

Examples

- The following command (1) can be used when the software is in on-chip variation analysis mode to add pessimism to both the setup and hold analyses by making the early paths 20 percent faster:

```
    setAnalysisMode -analysisType onChipVariation
    ...
(1) set_timing_derate -early 0.8 -late 1.0
```

- The following example uses two `set_timing_derate` commands to set the scaling factors independently for the min and max corners. When the software is in best-case worst-case analysis mode:
  - Command (1) adds pessimism to the setup analysis by making the latching clock path 20 percent faster.
  - Command (2) adds pessimism to the hold analysis by making the latching clock path 10 percent slower.

```
(1) set_timing_derate -max -early 0.8 -late 1.0
(2) set_timing_derate -min -early 1.0 -late 1.1
```

- When the software is in on-chip variation analysis mode, there is only one active delay corner; therefore, the `-min` and `-max` parameters should not be used. The following command sequence will generate a warning:

```
setAnalysisMode -analysisType onChipVariation
...
set_timing_derate -min -early 0.8 -late 0.9
set_timing_derate -max -early 1.1 -late 1.2
```

  In this situation, the software uses `0.8` for derating early paths, and `1.2` for derating late paths.

- The following command sets the early path scaling factor of 0.85 for instance `MULT_9` cell delays:

```
set_timing_derate -early -cell_delay 0.85 [get_cells {MULT_9}]
```

- The following command sets the early path scaling factor of 0.75 for the net named `MUX2/Z`:

```
set_timing_derate -net_delay -early 0.75 [get_nets {MUX2/Z}]
```

Related Information

reportTimingDerate

Related Topics

- mmmc.tcl in the *Encounter Flat Implementation Flow Guide*
- mmmc.tcl in the *Encounter Hierarchical Implementation Flow Guide*
- timing.tcl in the *Encounter Flat Implementation Flow Guide*
- timing.tcl in the *Encounter Hierarchical Implementation Flow Guide*

setTimingLibrary

```
setTimingLibrary
     [-max {Max | Min}]
     [-min {Max | Min}]
```

Sets the timing libraries to be used in setup analysis (late-path delay calculation) or hold analysis (early-path delay calculation).

The default setting, if this command is not used, is that only maximum libraries will be used for setup analysis (late-path delay calculation) and only minimum libraries will be used for hold analysis (early-path delay calculation).

Use this command only after loading all the timing libraries.

Important

You cannot use this command when the software is in multi-mode multi-corner (MMMC) mode. Instead, use the `create_delay_corner` command to set the timing libraries.

Parameters

`-max {Max | Min}`	Specifies the Max or Min libraries to be used in setup analysis.
`-min {Max | Min}`	Specifies the Max or Min libraries to be used in hold analysis.

Examples

- The following command sets the timing library to `Max` used for both setup and hold analysis (early-path delay calculation):

  ```
  setTimingLibrary -max Max -min Max
  ```

- The following script shows how to set a Min library in setup analysis and a Max library in hold analysis for reporting the detail data paths that do not meet timing in a report file:

  ```
  setTimingLibrary -max Min -min Max
  setAnalysisMode -checkType setup
  report_timing -max_points 10 > setupTiming.rpt
  setAnalysisMode -checkType hold
  report_timing -max_points 10 > holdTiming.rpt
  ```

  - The `setTimingLibrary -max Min -min Max` command sets the Min libraries for setup analysis and the Max libraries for hold analysis.

- The `setAnalysisMode` -checkType `setup` command sets the timing analysis mode to `setup`.
- The `report_timing -max_points 10 > setupTiming.rpt` command runs setup analysis using the `Min` Libraries and writes the result to the `setupTiming.rpt` file.
- The `setAnalysisMode` -`checkType hold` command sets the timing analysis mode to `hold`.
- The `report_timing -max_points 10 > holdTiming.rpt` command runs `hold` analysis using `Max` Libraries and writes the output to the `holdTiming.rpt` file.

setAnalysisMode

```
setAnalysisMode
     [-help]
     [-reset]
     [-analysisType {single | bcwc | onChipVariation}]
     [-asyncChecks {async | noAsync | asyncOnly}]
     [-caseAnalysis {true | false}]
     [-checkType {setup | hold}]
     [-clkNetsMarking {beforeConstProp | afterConstProp}]
     [-clkSrcPath {true | false}]
     [-clockGatingCheck {true | false}]
     [-clockPropagation {sdcControl | forcedIdeal | autoDetectClockTree}]
     [-cppr {none | both | setup | hold}]
     [-enableMultipleDriveNet {true | false}]
     [-honorActiveLogicView {true | false}]
     [-honorClockDomains {true | false}]
     [-log {true | false}]
     [-propSlew {true | false}]
     [-sequentialConstProp {true | false}]
     [-skew {true | false}]
     [-timeBorrowing {true | false}]
     [-timingEngine {statistical | static | pathBasedSSTA}]
     [-timingSelfLoopsNoSkew {true | false}]
     [-usefulSkew {true | false}]
     [-useOutputPinCap {true | false}]
     [-warn {true | false}]
```

Sets global analysis modes for timing analysis. The software uses these modes for all timing analysis commands unless you specify specific modes in the reporting commands. The modes that you specify in the reporting commands override these modes.

When you use the setAnalysisMode command, the software sets the following defaults:

- Sets the constants to be propagated while building the timing graph. The software reads the constants from the timing constraints file or the netlist.
- Sets the analysis mode to accept the module-level timing constraints of the internally specified *set_input_delay* and *set_output_delay* variables for the I/Os of the module for running timing analysis and timing optimization.

  The software applies the `set_input_delay` command and the `set_output_delay` command on internal pins during timing analysis.
- Treats the latches as latches.
- Reports the end points as many times as the number clocks that control that end point.

Important

Running "`timeDesign -postRoute -si [-hold]`" with "`setAnalysisMode -analysisType bcwc`" might lead to incorrect timing analysis results. This is because the SI delay push-outs (or pull-ins for -hold mode) on the clock nets get annotated to both the launch and the capture clock paths resulting in optimistic slacks to be reported. To get accurate timing analysis results, use "`setAnalysisMode -analysisType onChipVariation`", which requires the design to be run either in multi-mode multi-corner (MMMC) or single-mode single-corner (SMSC) mode. For more information, see the Specifying the MMMC Environment section in the Optimizing Timing chapter of the *Encounter User Guide*.

Parameters

`-analysisType {single | bcwc | onChipVariation}`

Sets the timing analysis type to single, best case worst case or on-chip variation.

Default: If you read in one library and do not specify any timing analysis type, the software uses `single` by default. If you read in two libraries and do not specify any timing analysis type, the software uses `bcwc` by default.

`single` Scales the delay values based on one operating condition. For more information, see "Single Timing Analysis Mode," in the Encounter User Guide.

`bcwc` Checks the design for two extreme operating conditions. The software uses the maximum delays for all paths during setup checks and minimum delays for all paths during hold checks. For more information, see "Best-Case Worst-Case (BC-WC) Timing Analysis Mode," in the Encounter User Guide.

`onChipVariation`

Calculates the delay for one path based on maximum operating condition while calculating the delay for another path based on minimum operating condition for setup or hold checks. For more information, see "On-Chip Variation (OCV) Timing Analysis Mode," in the Encounter User Guide.

`-asyncChecks {async | noAsync | asyncOnly}`

Reports timing violations for asynchronous timing checks, including recovery and removal, and other checks on pins defined as preset or clear in the Liberty file.

Default: `async`

`async`	Reports timing checks on pins that are defined as preset or clear in the Liberty file.
`noAsync`	Reports timing violations while ignoring the two time arc attributes.
`asyncOnly`	Reports only recovery and removal checks.

`-caseAnalysis {true | false}`

Determines whether constants supplied from the constraint file and netlist should be applied to the analysis.

Specifying `-caseAnalysis false` parameter blocks the application of constants from the timing constraints file. However, constants that are introduced through the netlist and timing library are still applied and propagated.

Default: `true`

`-checkType {setup | hold}`

Checks the design for setup or hold violations in the current analysis.

Default: `setup`

`-clkNetsMarking {beforeConstProp | afterConstProp}`

Note: This parameter is not available in CTE timing analysis mode.

Marks clock nets before or after propagating the constants.

Default: `markClkNetsBeforeConstProp`

`-clkSrcpath {true | false}`

Reports combinational paths from the clock source to the timing checks on data pins.

Note: If `-clkSrcPath false` is specified and the clock portion of the path is considered as ideal (due to the SDC constraints or the `-clockPropagation forcedIdeal` parameter setting), the delay calculation software treats the clock nets as ideal nets when calculating the delays and slews along the data path.

For all pre-clock tree synthesis (CTS), `optDesign`, and `timeDesign` steps, the software uses `-clkSrcPath false` in combination with `-clockPropagation forcedIdeal` to calculate the delays and slews along the data path.

For all post-CTS flow steps, the software uses the default `-clkSrcPath true` in combination with `-clockPropagation sdcControl` to calculate the delays and slews along the data path.

Default: `true`

`-clockGatingCheck {true | false}`

Reports the gating checks.

Default: `true`

`-clockPropagation {sdcControl | forcedIdeal | autoDetectClockTree}`

Specifies how to determine whether clock end points are interpreted as having either ideal or propagated clock timing.

Default: `sdcControl`

`sdcControl` Reports the timing violations considering clock tree and enables propagating clocks to report clock latency. You must set the `set_propagated_clock` constraint to propagate the clocks.

`forcedIdeal` Does not report the timing violations considering the clock tree.

`autoDetectClockTree`

Note: The `autoDetectClockTree` parameter is obsolete. The timing system will revert to `sdcControl`, if this parameter is specified.

`-cppr {none | both | setup | hold}`

Removes pessimism from clock paths that have a portion of the clock network in common between the clock source and clock destination paths.

The pessimism is introduced when the timing analysis tools assume that the common path has different delay values for two different paths in case of on-chip variation. The pessimism can also be because of clock reconvergence in the clock network. The pessimism effects all analysis modes including single, best case-worst case (`bcwc`), and on-chip variation.

For more information, see "Removing Common Path Pessimism", in the Encounter User Guide.

Note: If you use the `-cppr` parameter to do optimization, the run time might increase slightly.

`none` Disables removal of clock reconvergence pessimism.

Default: `none`

`both` Enables removal of clock reconvergence pessimism for both setup and hold modes.

When `setAnalysisMode -cppr` is specified (without any arguments), the value is set to `both`.

`setup` Enables removal of clock reconvergence pessimism in setup mode only.

`hold` Enables removal of clock reconvergence pessimism in hold mode only.

`-enableMultipleDriveNet {true | false}`

Enables the multiple-driver support for reporting.

Default: `true`

`-help`

Outputs a brief description that includes type and default information for each `setAnalysisMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setAnalysisMode`.

`-honorActiveLogicView {true | false}`

Sets the timing analysis mode to honor Active Logic View (previously referred to as CDTV and Virtual Partition). Instead of analyzing the whole chip, the software analyzes only the nets related to interface paths between the partitions.

The `createActiveLogicView` command sets `-honorActiveLogicView` by default, so there is no need to specify `setAnalysisMode -honorActiveLogicView true` explicitly after specifying `createActiveLogicView`.

Note: This parameter applies to pre-CTS mode only.

Default: `false`

`-honorClockDomains {true | false}`

Controls whether the software uses clock domains or path groups for timing optimization.

When set to `true`, `optDesign` uses standard and user-specified clock domains during timing optimization. The following clock domain related commands are enabled: `setClockDomains`, `clearClockDomains`, and `reportClockDomains`. This setting is backward compatible with the behavior of releases prior to 7.1.

When set to `false`, `optDesign` uses path groups during timing optimization. Clock domain related commands are disabled, and any active clock domains are cleared. Path groups can be created and modified using the following commands: `group_path`, `reset_path_group`, `createBasicPathGroups`, `setPathGroupOptions`, `resetPathGroupOptions`, `reportPathGroupOptions`. If no path groups are explicitly created, the software acts as though the basic path groups were created. However, as soon as you create one path group, the basic groups are then cleared.

Note: Path groups and clock domains are mutually exclusive: enabling one methodology disables the other.

Default: `true`

`-log {true | false}` Writes warning messages to the log file.

Default: `true`

`-propSlew {true | false}`

Propagates slew through the interconnect, when in hold analysis mode. Specify a value of `false` to ensure that the slew at the input of the receivers is not greater than the slew at the output of the drivers.

Default: `false`

`-reset` Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setAnalysisMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-sequentialConstProp {true | false}`

Propagates constants through sequential elements. If you specify `-sequentialConstProp false`, constants will stop at sequential elements.

Default: `false`

`-skew {true | false}`

Reports clock skew. The `-skew false` setting treats all clock latency as 0.

Default: `true`

With the `-skew true` setting, you can specify the `-clockPropagation` parameter.

- Using the default `-skew true -clockPropagation sdcControl` setting accounts for the clock latency during timing analysis and proceeds as follows:
  - If the `set_propagated_clock` command is specified in the timing constraint file, the clock latency is computed from the netlist (propagated mode).
  - If the `set_clock_latency` command is specified in the timing constraint file, the clock latency specified in the constraint is used.

Otherwise, no latency is reported (ideal mode)

`-timeBorrowing {true | false}`

Considers time borrowing during timing analysis. Time borrowing is the amount of time borrowed by a previous logic.

Default: `true`

`-timingEngine {statistical | static | pathBasedSSTA}`

Specifies the type of timing analysis to perform.

Default: `static`

`statistical` Sets to analysis mode to statistical static timing analysis (SSTA).

`static` Performs static timing analysis.

`pathBasedSSTA`

Note: This parameter is obsolete.

`-timingSelfLoopsNoSkew {true | false}`

Eliminates clock skew due to clock uncertainty for a path starting and ending at the same register. If the clock skew is not eliminated, the timing for such paths is pessimistic.

Note: Use this parameter before building the timing graph.

Default: `false`

`-usefulSkew {true | false}`

Considers the latency file (*`latency_file`*`.sdc`) during timing analysis. The latency file is generated when you run the `skewClock` command.

Default: `false`

`-useOutputPinCap {true | false}`

Includes the output pin capacitance when calculating delay values. When you specify `-useOutputPinCap false`, the software excludes only the pin capacitance of the driver for which you are calculating the delay.

Default: `true`

`-warn {true | false}`

Displays warning messages as the timing constraints file is read.

Default: `true`

Examples

- The following command sets timing analysis to report clock skew, suppresses warning messages, and check for setup timing:

  ```
  setAnalysisMode -checkType setup -skew true -warn false
  ```

- The following commands set the mode for timing analysis to consider latency files, and generate a timing report based on timing with latencies from pre-CTS useful skew:

  ```
  setAnalysisMode -usefulSkew true
  restoreDesign mydesign
  report_timing
  ```

- The following commands sets the mode for timing analysis to ignore latency files, and generate a timing report based on timing without latencies from pre-CTS useful skew:

```
setAnalysisMode -usefulSkew false
report_timing
```

- The following command reset the `-checkType` parameter to its default value:

```
setAnalysisMode -reset -checkType
```

- The following command resets all of the `setAnalysisMode` parameters to their default values:

```
setAnalysisMode -reset
```

Related Topics

- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Partitions in the *Encounter Hierarchical Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Top-Level in the *Encounter Hierarchical Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*

setClockDomains

```
setClockDomains
     [-fromType {input | register | flopOrLatch}]
     [-toType {output | register | flopOrLatch}]
```

Sets the clock domains for timing analysis. You can use the command incrementally to set more than one clock domain for a timing analysis session.

To optimize a single clock domain, you must clear the clock domains that you had set previously and set the clock domain that you need to optimize. To clear the clock domains, use the `clearClockDomains` command.

Parameters

`-fromType {input | register | flopOrLatch}`

Includes paths based on the source point of paths.
Default: All source points are used

The source point can be one of the following:

`input`	Includes paths that originate at the inputs.
`register`	Includes paths that originate at the registers and macros.
`flopOrLatch`	Includes paths that originate at flip-flops and latches.

`-toType {output | register | flopOrLatch}`

Includes paths based on the destination points of paths.
Default: All destination points are used

The destination point can be one of the following:

`output`	Includes paths that terminate at the outputs.
`register`	Includes paths that terminate at the registers and macros.
`flopOrLatch`	Includes paths that terminate at flip-flops and latches.

Command Order

Use this command before running timing analysis.

Examples

- The following command reports path delays from register to register:

```
setClockDomains -fromType register -toType register
```

- The following script creates and reports timing based on specific path groups. These commands set up three path groups: flop-2-flop, input-2-flop, and flop-2-output and shows the ten worst paths in all three path groups:

```
clearClockDomains
setClockDomains -fromType register -toType register
report_timing
clearClockDomains
setClockDomains -fromType register -toType output
report_timing
clearClockDomains
setClockDomains -fromType input -toType register
report_timing
clearClockDomains
```

Related Topics

- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Partitions in the *Encounter Hierarchical Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Top-Level in the *Encounter Hierarchical Implementation Flow Guide*
- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- SI Closure in the *Encounter Flat Implementation Flow Guide*

- Partition SI Closure in the *Encounter Hierarchical Implementation Flow Guide*
- Top-Level SI Closure in the *Encounter Hierarchical Implementation Flow Guide*

setTimingDerate

```
setTimingDerate
     [-early value]
     [-late value]
     [-min | -max | -delay_corner delayCornerName]
     [-clock]
     [-data]
     [-net_delay]
     [-cell_delay]
     [-retain_delay]
     [-cell_check]
```

Important

Command "`setTimingDerate`" is obsolete and has been replaced by "`set_timing_derate`".

Sets delay scaling or derating factors for early and late paths in the current design. You use this command to perform scaled on-chip variation around best-case and worst-case corners. The timing scaling factors affect the delay values generated in the timing reports. You can set scaling factors for data paths, clock paths, minimum and maximum operating conditions.

If you do not specify the `-clock` or `-data` parameter, the software applies the scaling factors to both clock and data paths. Similarly, if you do not specify the `-min` or `-max` parameter, the software applies the scaling factor to both minimum and maximum operating conditions.

If you specify only one of the two parameters, `-data` or `-clock`, the software uses the last defined value for the other.

You use the `reportTimingDerate` command to view the derating factor settings.

Note: If you use the `setTimingDerate` command, you must not use the `set_timing_derate` constraint in the constraint file to set the derating factor.

Parameters

`-cell_check`	Specifies scaling on timing checks. Use the `-cell_check` option with the `-early` option to specify scaling on hold arcs and use with the `-late` option to specify scaling on setup arcs.
`-cell_delay`	Specifies scaling on cell delays. If neither the `-cell_delay` nor the `-net_delay` option is specified, the specified scaling factor applies to both types of delays.

`-clock`	Applies the derating factors to clock paths only.
`-data`	Applies the derating factors to data paths only.
`-delay_corner` *`delayCornerName`*	Applies the derating factor to the specified delay calculation corner.
`-early` *`value`*	Specifies the derating factor for early paths. Early paths are launching clock during hold analysis, receiving clock during setup analysis and datapath during hold analysis. For definition of early and late paths, see "Definition of Early Late Paths" in *Encounter User Guide*. *Default:* `1.0`
`-late` *`value`*	Specifies the derating factor for late paths. Late paths are receiving clock during hold analysis, launching clock during setup analysis and datapath during setup analysis. *Default:* `1.0`
`-min`	Applies the derating factor to the minimum operating condition.This parameter is ignored during on-chip variation analysis. **Note:** You cannot use the `-min` parameter when the software is in multi-mode multi-corner (MMMC) mode. You must use the `-delay_corner` parameter instead.
`-max`	Applies the derating factor to the maximum operating condition. This parameter is ignored during on-chip variation analysis. **Note:** You cannot use the `-max` parameter when the software is in multi-mode multi-corner (MMMC) mode. You must use the `-delay_corner` parameter instead.
`-net_delay`	Specifies scaling on net delays.
`-retain_delay`	Specifies scaling on retain constructs. If you specify both `-retain_delay` and `-cell_delay` parameters, the software uses the retain_delay value for retain construct scaling. If you do not specify the `-retain_delay` parameter, retain constructs are scaled in the same manner as cell delays.

Command Order

Use this command after running clock tree synthesis and setting the timing analysis to propagated clock mode.

Examples

- The following command (1) can be used when the software is in on-chip variation analysis mode to add pessimism to both the setup and hold analyses by making the early paths 20 percent faster:

```
    setAnalysisMode -analysisType onChipVariation
    ...
(1) setTimingDerate -early 0.8 -late 1.0
```

- The following example uses two `setTimingDerate` commands to set the scaling factors independently for the min and max corners. When the software is in best-case worst-case analysis mode:
  - Command (1) adds pessimism to the setup analysis by making the latching clock path 20 percent faster.
  - Command (2) adds pessimism to the hold analysis by making the latching clock path 10 percent slower.

```
(1) setTimingDerate -max -early 0.8 -late 1.0
(2) setTimingDerate -min -early 1.0 -late 1.1
```

- When the software is in on-chip variation analysis mode, there is only one active delay corner; therefore, the `-min` and `-max` parameters should not be used. The following command sequence will generate a warning:

```
setAnalysisMode -analysisType onChipVariation
...
setTimingDerate -max -early 0.8 -late 1.0
setTimingDerate -min -early 1.0 -late 1.1
```

  In this situation, the software simply drops the `-max`, or `-max` qualifier, and applies the constraint. In the above example, the second `setTimingDerate` command would override the first one.

timeDesign

```
timeDesign
     {-prePlace | -preCTS | -postCTS | -postRoute [-si] | -signoff[-si] |
      -reportOnly}
     [-idealClock]
     [-drvReports]
     [-slackReports]
     [-pathreports]
     [-timingDebugReport]
     [-hold]
     [-expandReg2Reg]
     [-numPaths number_of_detailed_paths]
     [-prefix file_name_prefix]
     [-ilm]
     [-expandedViews]
     [-useTransitionFiles]
     [-outDir out_dir]
```

Runs Trial Route, extraction, and timing analysis, and generates detailed timing reports. The generated timing reports are saved in `./timingReports` directory or the directory that you specify using the `-outDir` parameter.

The following timing reports are generated:

- Setup and hold time for the following path groups:
  - Register to register (*TopCellName_DesignStage*`_reg2reg.tarpt`)
  - Input to register (*TopCellName_DesignStage*`_in2reg.tarpt.` This report also includes the clock source paths (if applicable to the design). The clock source paths are included even if the clock source is an internal pin.
  - Register to output (*TopCellName_DesignStage*`_reg2out.tarpt`)
  - Input to output (*TopCellName_DesignStage*`_in2out.tarpt`)
  - All paths ending in clock gate (*TopCellName_DesignStage*`_clkgate.tarpt`).

  For hold analysis, the report names are *TopCellName_DesignStage_PathGroup_hold*`.tarpt.`

Note: If you use the `setClockDomain` command to set a clock domain, the software generates one report for that path group instead of generating four for all path groups.

- Setup and hold time without using path groups:
  - Setup analysis (*TopCellName_DesignStage*`_all.tarpt`)

- Hold analysis (*TopCellName_DesignStage*_all_hold.tarpt)

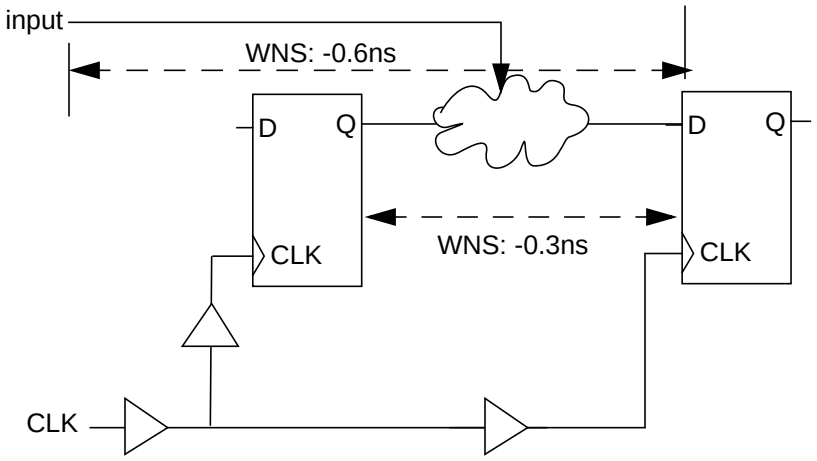
- Density

  Note: The density value reported by the timeDesign command might differ from the effective utilization value reported by the checkPlace command because timeDesign does not consider padding when calculating density, but checkPlace does consider it when calculating effective utilization.

- Maximum transition time violation (*TopCellName_DesignStage*.tran)
- Maximum capacitance violation (*TopCellName_DesignStage*.cap)
- Maximum fanout violation (*TopCellName_DesignStage*.fanout)
- Glitch violations in the *outDir*/*prefix_hold*/setup directory.
- Summary (*TopCellName_DesignStage*.summary and *TopCellName_DesignStage*_hold.summary)

The overall total negative slack (TNS) and number of violating paths of a design might not be equal to the total of the TNS and violating paths of the individual path groups. This is because the TNS and number of violating paths are based on end-point of the path and are not path based.

For example, the following figure has a register with two paths, one from a primary input with a slack of -0.6ns and other from another register with a slack of -0.3ns.



In that case the overall TNS will be -0.6ns with 1 violating path (end point-based). But the individual reg2reg TNS will be -0.3ns with 1 violating path and in2reg with a TNS of -0.6ns with 1 violating path. Therefore, the sum total of individual path group TNS is not the same as overall TNS.

Similarly, when you run the `timeDesign` command while performing multi-mode multi-corner analysis, overall TNS is not the sum of TNS of each view considered separately. The command sums the worst-case slack at each endpoint across all views. Therefore each endpoint contributes only in overall worst-slack in the calculation of TNS.

Important

Running "`timeDesign -postRoute -si [-hold]`" with "`setAnalysisMode -analysisType bcwc`" might lead to incorrect timing analysis results. This is because the SI delay push-outs (or pull-ins for -hold mode) on the clock nets get annotated to both the launch and the capture clock paths resulting in optimistic slacks to be reported. To get accurate timing analysis results, use "`setAnalysisMode -analysisType onChipVariation`", which requires the design to be run either in multi-mode multi-corner (MMMC) or single-mode single-corner (SMSC) mode. For more information, see the Specifying the MMMC Environment section in the Optimizing Timing chapter of the *Encounter User Guide*.

Parameters

`-drvReports`	Generate design rule violation (DRV) reports only. *Default:* Generates DRV, slack, and path reports.
`-expandedViews`	Generates detailed view-specific timing reports when the software is in multi-mode multi-corner analysis mode. For multi-mode multi-corner analysis, the software creates a separate directory for each view. For example, if your design has two analysis views, `view1` and `view2`, the output reports are generated in the `./timingReports/view1` and `./timingReports/view2` directories.
`-expandReg2Reg`	Specifies that the register to register path groups are divided into the following detailed path groups for generating reports: ■ Flop to flop ■ Macro to flop ■ Flop to macro ■ Macro to macro

`-hold`
Reports hold violations only. Before ending the `timeDesign` run, the software reverts back to the initial analysis mode that was set by the `setAnalysisMode` command (either setup or hold).

Default: Reports only setup violations

`-idealClock`
Performs ideal clock analysis irrespective of the design stage.

Default: The software performs propagated clock analysis with the `-postCTS` and `-postRoute` parameters. The software performs ideal clock analysis with the `-prePlace` and `-preCTS` parameters.

Note: This parameter is not honored when used with `-reportOnly.`

`-ilm`
Note: The `-ilm` parameter is obsolete, because the command now detects ILMs by default.

Performs timing analysis on designs containing Interface Logic Models (ILMs).

This feature flattens ILMs, analyzes timing, then unflattens the ILMs. You can use this feature in pre-CTS mode only for setup checking. You can use this feature in post-CTS and post-route modes for setup and hold checking. The `timeDesign -postRoute -si` command checks for SI violations in addition to checking setup and hold violations. The `-signoff` mode is also supported with `-ilm`.

You cannot specify this parameter when the software is in multi-mode multi-corner analysis mode, because multi-mode multi-corner analysis is not supported in the hierarchical flow.

`-numPaths` *`num_paths`*

Specifies the number of paths to report in the detailed path violations.

Default: `50`

`-outDir` *`out_dir`*
Specifies the name of the output directory that contains all the generated reports.

Default: `timingReports`

`-pathReports`
Generate path reports only.

Default: Generates DRV, slack, and path reports.

`-preCTS` Generates timing reports before the clock tree is built. By default, in this mode the software runs Trial Route, native default extraction, and timing analysis for generating timing reports. The `-preCTS` parameter sets `setAnalysisMode` to `-clkSrcPath false` and `-clockPropagation forcedIdeal`.

`-prePlace` Generates timing reports before the design is placed. In this mode, the software ignores the net load while building the timing graph and runs timing analysis for generating timing reports. The DRV report files are not generated in this mode, and the DRV numbers are not reported in the summary for this parameter. The `-prePlace` parameter sets `setAnalysisMode` to `-clkSrcPath false` and `-clockPropagation forcedIdeal`.

> **Note:** The `-reportOnly` parameter is not honored when used with this parameter.

`-postCTS` Generates timing reports for a design whose clock tree has been created. By default, in this mode the software runs Trial Route, native default extraction, and timing analysis for generating timing reports. The `-postCTS` parameter sets `setAnalysisMode` to `-clkSrcPath true` and `-clockPropagation sdcControl`.

`-postRoute` Generates timing reports for a design whose routing is complete. By default, in this mode the software runs native detailed extraction, and timing analysis for generating timing reports. The `-postRoute` parameter sets `setAnalysisMode` to `-clkSrcPath true` and `-clockPropagation sdcControl`.

`-prefix` *`file_name_prefix`*

Specifies a prefix for the timing report file names.

Default: `DesignName_DesignStage`

`-reportOnly`

Specifies the use of existing extraction and timing analysis data to generate timing reports. When you use this parameter, the software does not run extraction or timing analysis; instead it uses data that is already in memory.

- `-reportOnly` is not honored when used with `-prePlace`.
- `-idealClock` is not honored when used with `-reportOnly`.
- `-signOff` is honored when used with `-reportOnly`. This is used to run a signoff timing analysis based on external SPEF files.

Note: Specifying `-preCts`, `-postCts`, or `-postRoute` is optional with `-reportOnly`.

`-si`

Generates glitch violation report and incremental SDF for timing analysis. This parameter also creates a `.cdb` model for a block.

You can use this parameter with the `-postRoute` and `-signoff` parameters only. You must ensure that the Encounter configuration file contains data for signal integrity analysis.

The `timeDesign` command considers the `setSIMode` command settings.

Note: You can analyze signal integrity problems in multiple CPU processing mode. For information on the commands to use, see the Multiple-CPU Processing Commands chapter.

`-signoff`

Generates timing reports for sign-off timing analysis. By default, in this mode the software runs SignalStorm delay calculation and QRC standalone sign-off extraction.

To use this parameter, you must ensure that the Encounter configuration file contains data for QRC standalone sign-off extraction.

When the software is in multi-mode multi-corner analysis mode, this parameter calls RC extraction multiple times for each analysis view to generate and annotate the SPEF files for each RC corner.

Note: You can use the `-reportOnly` parameter in addition to `-signOff` to run a signoff timing analysis based on external SPEF files.

`-slackReports`	Generate slack reports only. *Default:* Generates DRV and path reports.
`-timingDebugReport`	Generates a compressed (`.gz`) detailed timing report in ASCII format. This report is used for debugging timing results using the timing debug feature in Encounter. For more information on debugging timing results, see "Debugging Timing Results" in Encounter User Guide.
`-useTransitionFiles`	Reports the maximum transition violations when using external transition files specified by the `readTransitionFile` command.

Command Order

Use this command after loading the design.

Examples

- The following command runs Trial Route, native default extraction, and timing analysis to generate timing reports before clock tree synthesis:

  ```
  timeDesign -preCTS
  ```

- The following example shows a summary report generated by `timeDesign`:

```
+--------------------+---------+---------+---------+---------+---------+---------+
|     Setup mode     |   all   | reg2reg | in2reg  | reg2out | in2out  | clkgate |
+--------------------+---------+---------+---------+---------+---------+---------+
|           WNS (ns):| -0.491  | -0.491  |   N/A   |  6.868  | N/A     |   N/A   |
|           TNS (ns):|-866.856 |-866.856 |   N/A   |  0.000  | N/A     |   N/A   |
|    Violating Paths:|  5273   |  5273   |   N/A   |    0    | N/A     |   N/A   |
|          All Paths:|  69372  |  69343  |   N/A   |   29    | N/A     |   N/A   |
+--------------------+---------+---------+---------+---------+---------+---------+

+----------------+----------------------+---------+
|                |        Real          |  Total  |
|    DRVs        +---------+------------+---------|
|                |    Nr   | Worst Vio  |    Nr   |
+----------------+---------+------------+---------+
|   max_cap      |    3    |   -0.004   |    4    |
|   max_tran     |    0    |   0.000    |    0    |
|   max_fanout   |    0    |     0      |    0    |
+----------------+---------+------------+---------+

Density: 53.455%
Routing Overflow: 0.00% H and 0.00% V
------------------------------------------------------------
```

- The following command ignores the net load while building the timing graph and runs timing analysis for generating timing reports before the design is placed:

  ```
  timeDesign -prePlace
  ```

- The following commands run native detailed extraction, and timing analysis to generate timing reports for both setup and hold violations after the design has been routed:

  ```
  timeDesign -postRoute
  timeDesign -postRoute -hold
  ```

- The following command runs QRC extraction, and SI analysis for final signoff of the design:

  ```
  timeDesign -signoff -si
  ```

- The following command generates timing reports in multi-mode multi-corner timing analysis view, where test1 is the specified analysis view:

  ```
  set_analysis_view -setup test1 -hold test1
  timeDesign -postRoute -outDir sample
  ```

The following summary report is generated:

timeDesign Summary						
Setup mode	all	reg2reg	in2reg	reg2out	in2out	clkgate
WNS (ns):	-10.821	-7.001	-7.264	-4.978	-10.821	N/A
TNS (ns):	-6898.5	-5609.5	-1528.6	-55.664	-150.330	N/A
Violating Paths:	4836	2756	2891	16	16	N/A
All Paths:	6866	6698	5654	84	16	N/A
test1	-10.821	-7.001	-7.264	-4.978	-10.821	N/A
	-6898.5	-5609.5	-1528.6	-55.664	-150.330	N/A
	4836	2756	2891	16	16	N/A
	6866	6698	5654	84	16	N/A
Density: 69.829%						
Real DRV (fanout, cap, tran): (2, 524, 104)						
Total DRV (fanout, cap, tran): (10, 524, 104)						

Related Topics

- Load and Check Data in the *Encounter Flat Implementation Flow Guide*
- Signoff in the *Encounter Flat Implementation Flow Guide*
- “Accelerating the Design Process by Using Multiple-CPU Processing” chapter of the *Encounter User Guide*

unloadTimingCon

```
unloadTimingCon
```

Clears the timing constraints read during design import. The timing graph is also cleared. You can load another timing constraints file after using this command.

You cannot use this command when the software is in multi-mode multi-corner mode. Use the `create constraint mode` command to control the use of timing constraint files.

Parameters

None

Command Order

Use this command after importing the design.

update_analysis_view

```
update_analysis_view
     -name existingAnalysisViewName
     { -constraint_mode newModeName | -delay_corner newDelayCornerObj}
```

Changes attribute values for the specified existing analysis view.

Important

You can update an analysis view before the multi-mode multi-corner definition file is loaded into the design, or after. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

Parameters

`-constraint_mode` *`newModeName`*

Specifies the name of the new constraint mode to associate with this analysis view.

`-delay_corner` *`newDelayCornerObj`*

Specifies the name of the new delay calculation corner to associate with this analysis view.

`-name` *`existingAnalysisViewName`*

Specifies the name of the existing analysis view to update.

Examples

- The following command creates an analysis view called `missionSlow` using the constraint mode `missionSetup` and the delay calculation corner `dcWCCOM`:

  ```
  create_analysis_view
      -name missionSlow
      -constraint_mode missionSetup
      -delay_corner dcWCCOM
  ```

- The following command changes the delay calculation corner associated with the analysis view `missionSlow` to `dcMax`:

  ```
  update_analysis_view -name missionSlow -delay_corner dcMax
  ```

update_constraint_mode

```
update_constraint_mode
     -name existingConstraintModeName
     {-sdc_files {newFile1.sdc newFile2.sdc...}
     | -ilm_sdc_files {newFile1.sdc newFile2.sdc ...}}
```

Changes the SDC constraint file information for the specified existing constraint mode object.

Note: The `update_constraint_mode` command resets all interactive constraints. If you define interactive constraints and then use the `update_constraint_mode` command, the interactive constraints for all the views will be reset.

Important

You can update a constraint mode object before the multi-mode multi-corner definition file is loaded into the design, or after. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

Parameters

`-ilm_sdc_files {`*`newFile1`*`.sdc `*`newFile2`*`.sdc ...}`

Specifies the new SDC constraint files for the ILM flow to associate with this constraint mode.

`-name `*`existingConstraintModeName`*

Specifies the name of the existing constraint mode object to update.

`-sdc_files {`*`newFile1`*`.sdc `*`newFile2`*`.sdc ...}`

Specifies the new SDC constraint files to associate with this constraint mode.

Examples

- The following command groups the SDC files `io.sdc`, `mission1-clks.sdc`, and `mission1-except.sdc` to create a mode object named `missionSetup`:

```
create_constraint_mode -name missionSetup
    -sdc_files [list io.sdc mission1-clks.sdc mission1-except.sdc]
```

- The following command changes the SDC files associated with the constraint mode `missionSetup` to `io.sdc`, `mission3-clks.sdc`, and `mission3-except.sdc`:

```
update_constraint_mode -name missionSetup
    -sdc_files {io.sdc mission3-clks.sdc mission3-except.sdc}
```

Related Topics

- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

update_delay_corner

```
update_delay_corner
     -name delayCornerObj
     { -power_domain powerDomainName
       { -library_set libSetObj
         [-opcond_library libName]
         [-opcond opcondName]
         [-irdrop_file list_of_files]

       | -late_library_set libSetName
         -early_library_set libSetName
         [-late_opcond_library libName]
         [-early_opcond_library libName]
         [-late_opcond opcondName]
         [-early_opcond opcondName]
         [-late_irdrop_file list_of_files]
         [-early_irdrop_file list_of_files]
       }
     }
     |
     { [-library_set libSetObj]
       [-opcond_library libName]
       [-opcond opcondName]
       [-rc_corner rcCornerObj]
       [-irdrop_file list_of_files]

     | [-late_library_set libSetName]
       [-early_library_set libSetName]
       [-late_opcond_library libName]
       [-early_opcond_library libName]
       [-late_opcond opcondName]
       [-early_opcond opcondName]
       [-late_irdrop_file list_of_files]
       [-early_irdrop_file list_of_files]
     }
```

Modifies the parameters of an existing delay calculation corner object. You can use the `update_delay_corner` command to add or change attribute values for an existing delay calculation corner object, or for an existing power domain definition.

You can update a delay calculation corner object before the multi-mode multi-corner definition file is loaded into the design, or after. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

You also can use the `update_delay_corner` command to add a power domain definition to the specified delay calculation corner. Instances contained within an MSMV power domain often require a different library or operating condition that used for the default power domain.

Power domain definitions within a delay calculation corner object can be used to assign specific libraries and PVT settings per power domain.

When the software is not in multi-mode multi-corner mode, the binding of timing libraries and operating conditions to power domains are specified using `createPowerDomain -minTimingLib` and `-maxTimingLib`, and `setOpCond -powerDomain` parameters. When the software is in multi-mode multi-corner mode, this use model is disabled.

Important

You must use the same power domain names that you intend to specify when using the `createPowerDomain` command to define the physical aspects of the domain.

Note: Use separate delay calculation corners to define major PVT corner differences. Use the `-early_*` and `-late_*` parameters within a single delay calculation corner to control on-chip-variation.

The `-early_*` and `-late_*` parameters can be specified separately, to update a delay corner object with early or late information. If you only specify one of the `-early_*` or `-late_*` parameters, the software uses the original information pointer in place of the missing early/late parameter.

For example, if you initially created a delay corner object using:

```
create_delay_corner
    -library_set libsNominal
    ...
```

If you specify `update_delay_corner -early_library_set libsEarly`, the software actually uses:

```
update_delay_corner
    -early_library_set libsEarly
    -late_library_set libsNominal
```

Use this command after creating at least one delay calculation corner (`create_delay_corner`).

Parameters

`-early_irdrop_file` *`list_of_files`*

Specifies the list of IR drop files to apply when calculating early arrival times at a single power domain.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

`-early_library_set` *`libSetName`*

Specifies the library set to associate with the power domain for calculating early arrival times for this power domain.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

`-early_opcond` *`opcondName`*

Specifies the operating condition to use for calculating early arrival times for this power domain.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

Default: Each library in the early library set uses its own default operating condition.

`-early_opcond_library` *`libName`*

Specifies the internal library name for the library in which the early operating condition is defined. This is *not* the library file name.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

Default: The software searches the early library set for the specified operating condition (`-early_opcond`), starting with the master library.

`-irdrop_file` *`list_of_files`*

Specifies the list of IR drop files to apply to both early and late delay calculation for this power domain object.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

`-late_irdrop_file` *`list_of_files`*

Specifies the list of IR drop files to apply when calculating late arrival times at a single power domain.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

`-late_library_set` *`libSetName`*

Specifies the library set to associate with this delay corner object for calculating late arrival times for this power domain.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

`-late_opcond` *`opcondName`*

Specifies the operating condition to use for calculating late arrival times for this power domain.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

Default: Each library in the late library set uses its own default operating condition.

`-late_opcond_library` *`libName`*

Specifies the internal library name for the library in which the late operating condition is defined. This is *not* the library file name.

This parameter is used primarily when configuring the multi-mode multi-corner environment for on-chip-variation (OCV) analysis.

Default: The software searches the late library set for the specified operating condition (`-late_opcond`), starting with the master library.

`-library_set` *`libSetName`*

Specifies the library set to associate with the power domain. All of the cells and macros in the power domain get their timing and power information from these libraries.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

`-name` *`dcCornerObj`*

Specifies the delay calculation corner to which to add the power domain definition, or to which to update the delay corner or power domain attribute values.

`-opcond` *`OpcondName`*

Specifies the operating condition to use for the power domain.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

Default: Each library in the library set uses its own default operating condition.

`-opcond_library` *`libName`*

Specifies the internal library name for the library in which the operating condition is defined. This is *not* the library file name.

This parameter is used primarily to configure single-corner or Best-Case Worst-Case (BC-WC) analysis modes.

Default: The software searches the library set for the specified operating condition (`-opcond`), starting with the master library.

`-power_domain` *`powerDomainName`*

Specifies the name of the power domain to add to the delay calculation corner. This name must match the name of a power domain specified with the `createPowerDomain` command.

`-rc_corner` *`rcCornerObj`*

Specifies the new RC corner object to associate with this delay corner.

Examples

- The following command creates the delay calculation corner `dc1`:

```
create_delay_corner -name dc1
```

```
    -library_set libs-2volt
    -opcond_library delayvolt_2V
    -opcond slow_2V
    -rc_corner rc-cworst
```

- The following commands add definitions for the power domains `domain-3V` and `domain-5V` to the `dc1` delay calculation corner:

```
update_delay_corner -name dc1
    -power_domain domain-3V
    -library_set libs-3volt
    -opcond_library delayvolt_3V
    -opcond slow_3V
update_delay_corner -name dc1
    -power_domain domain-5V
    -library_set libs-5volt
    -opcond_library delayvolt_5V
    -opcond slow_5V
```

- The following command changes the libraries used by the delay calculation corner `dc1` to those in the library set `IsCOM-2v`, and changes the RC corner associated with `dc1` to `rc-ctyp`:

```
update_delay_corner -name dc1
    -library_set IsCOM-2V
    -rc_corner rc-ctyp
```

update_library_set

```
update_library_set
     -name existingLibSetName
     {-timing {newLib1.lib newLib2.lib ...}
     | -si {newLib1.cdb newLib2.cdb ...} }
```

Changes timing and cdB library files for the specified existing library set.

Important

You can edit a library set before the multi-mode multi-corner view definition file is loaded into the design, or after. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

Parameters

`-name existingLibSetName`

Specifies the name of the existing library set to update.

`-si {newLib1.cdb newLib2.cdb ...}`

Specifies the new cdB libraries to include in this library set.

`-timing {newLib1.lib newlib2.lib ...}`

Specifies the new timing libraries to include in this library set.

Examples

- The following command creates a library set that associates timing libraries and cdB libraries with a nominal voltage of 1 volt with the library name `IsCOM-1V`:

```
create_library_set
    -name IsCOM-1V -timing [list stdcell_F_1V.lib ram_F.lib pad.lib]
    -si [list stdcell_F_2.cdb ram_F.cdb pad.cdb]
```

- The following command changes the timing libraries associated with the library set `IsCOM-1V` to `stdcell_CVF_1V.lib` `ram_CVF.lib` `pad.lib`:

```
update_library_set -name minLibs
    -si {stdcell_CVF_1V.lib ram_CVF.lib pad.lib}
```

update_rc_corner

```
update_rc_corner
     -name existingRcCornerName
     { -cap_table newCapTableFile]
     | -T newTempValue
     | -default_res_factor float
     | -detailed_res_factor float
     | -res_factor newFloatValue
     | -default_cap_factor newFloatValue
     | -detailed_cap_factor newFloatValue
     | -xcap_factor newFloatValue
     | -qx_lib_file newFileName
     | -qx_tech_file newFileName
     | -qx_conf_file newFileName
     }
```

Adds or changes attribute values for the specified existing RC corner object.

You can edit an RC corner object before the multi-mode multi-corner definition file is loaded into the design, or after. However, after the software processes the file, any change to an existing object when the software is in multi-mode multi-corner mode results in the timing, delay calculation, and RC data being reset.

Parameters

`-cap_table` *`newCapTableFile`*

Specifies a new capacitance table to use for extraction when using this RC corner.

`-default_cap_factor` *`newFloatValue`*

Specifies a new capacitance scale factor for RC extraction in default mode.

`-default_res_factor` *`float`*

Specifies a new resistance scale factor for correlation, when the software is using default extraction.

`-detailed_cap_factor` *`newFloatValue`*

Specifies a new capacitance scale factor for RC extraction in detailed mode.

`-detailed_res_factor` *`float`*

Specifies a new resistance scale factor for correlation, when the software is using detailed extraction.

`-name` *`existingRcCornerName`*

Specifies the name of the existing RC corner object to update.

`-qx_conf_file` *`newFileName`*

Specifies a new user-created configuration file to use with QRC gate-level extraction.

`-qx_lib_file` *`newFileName`*

Specifies a optional cell library file used by QRC extraction to perform sign-off RC extraction.

`-qx_tech_file` *`newFileName`*

Specifies a new QRC technology (`.tch`) file which is used to perform sign-off RC extraction.

`-res_factor` *`newFloatValue`*

Note: The `-res_factor` parameter is obsolete. The parameter still works in this release, but to ensure compatibility with future releases, use the `-default_res_factor` or `-detailed_res_factor` parameter instead.

Specifies a new resistance scale factor for correlation.

`-T` *`newTempValue`* Specifies a new temperature, in units of Celsius, to use to derate resistance values.

`-xcap_factor` *`newFloatValue`*

Specifies a new cross-coupling capacitance scale factor.

Examples

- The following command creates an RC corner called `rc-cbest` that uses the capacitance table `myTech_bc.CapTbl`, and derates the resistance values based on the temperature of 50 Celsius:

  ```
  create_rc_corner -name rc-cbest -cap_table myTech.4_bc.CapTbl -T 50
  ```

- The following command changes the capacitance table used by RC corner `rc-cbest` to `myCap.2_bc.capTbl`, and the temperature by which resistance values are derated to 25 Celsius:

  ```
  update_rc_corner -name rc_cbest
  ```

```
-T 25
```

write_global_slack_report

```
write_global_slack_report
     [-min_slack val]
     filename
```

Reports early and late slacks on every instance pin in the design.

Parameters

`-min_slack val`

If specified, only the pins that have early and late slacks less than the specified value are written out.

Type: Float

filename

Specifies the output file name.

Note: To write a compressed report, add the `.gz` extension to the file name.

Example

The following command reports the early and late slacks on every instance pin in the design:

```
write_global_slack_report out.rpt
```

Example 32-21 write_global_slack_report Output

```
# Unit: 1ps
#     <late_rise>     <late_fall>    <early_rise>    <early_fall> <hpin_name>
# ------------------------------------------------------------------------------
       -4176.76        -4049.26       -82696.70      -82823.75    1_pin_name
       -7450.33        -6944.72       -82284.92      -82297.52    2_pin_name
       -4128.03        -3988.92       -82745.17      -82883.96    3_pin_name
       -5683.80        -5937.10       -86789.41      -86535.58    4_pin_name
```

Related Information

write_global_slack_worst_trigger_path_on_clocks

write_sdf

```
write_sdf
     [-version  { 2.1 | 3.0 }]
     [-precision non_neg_integer]
     [-scale float]
     [-delimiter char]
     [-transform_out_to_out_arcs]
     [-celltiming {all | none | nochecks}]
     [-interconn {all | none | noport | noinport | nooutport}]
     [-edges {edged | library | noedge | check_edge}]
     [-remashold]
     [-splitrecrem]
     [-splitsetuphold]
     [-setuphold [merge_always | split | merge_when_paired]]
     [-recrem [merge_always | split | merge_when_paired]]
     [-condelse]
     [-nonegchecks]
     [-min_period_edges {posedge | negedge | both | none}]
     [-no_escape]
     [-merge_arcs]
     [-process]
     [-voltage]
     [-temperature]
     [-filter]
     [-resort_values_min_to_max]
     [-view viewName]
     [-min_view viewName]
     [-typ_view viewName]
     [-max_view viewName]
     [-collapse_internal_pins]
     file_name
```

Writes delays to a Standard Delay Format (SDF) file. The `write_sdf` command can generate both plain-text and GNU zipped (`.gz`) SDF files.

By default, the `write_sdf` command uses the delays already cached by the system and writes only values of the triplets. **Parameters**

`-celltiming {all | none | nochecks}`

Specifies which cell delays and timing checks to write out.
Default: `all`

- `all` – Writes all cell delays and timing checks to the SDF file. This is the default.
- `nochecks` – Disables the writing of timing checks only.
- `none` – Prevents cell delays and timing checks from being written into the SDF file.

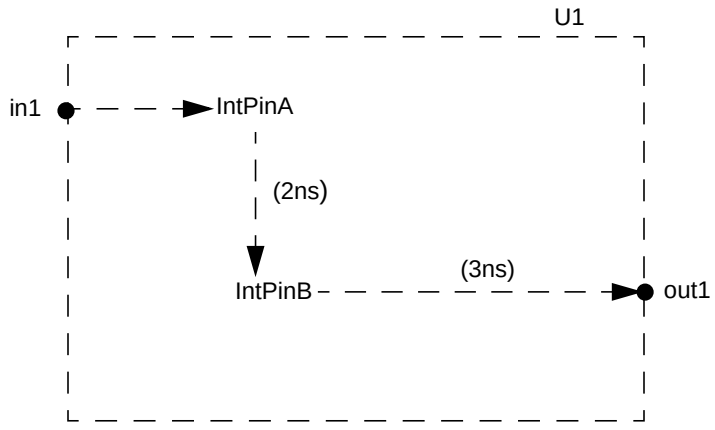
`-collapse_internal_pins`

Collapses arcs across internal pins to a single primary input to primary output arc.

Use the `-collapse_internal_pins` parameter when the SDF file being generated is intended for use by an application that does not support the same internal pin structure in its leaf cell libraries (for example, Cadence VXL and NCsim Verilog simulation products). Do *not* use this parameter to generate an SDF file intended for use within the Encounter flow, or by third party tools that utilize the same Liberty timing models.

Default: Arcs between internal pins in a timing model are written to the SDF file as explicit annotations to and from the internal pins internal pins.

For example, assume the following arc between internal pins:



The following SDF would be generated by default:

```
(U1/in1     U1/IntPinA  (1::1) (1::1))
(U1/IntPinA U1/IntPinB  (2::2) (2::2))
(U1/IntPinB U1/out1     (3::3) (3::3))
```

If you specify `-collapse_internal_pins`, the resulting SDF would show a single arc with the merged delays:

```
(U1/in1 U1/out1     (6::6) (6::6))
```

`-condelse`

Writes `CONDELSE` constructs with the default value when a `COND` construct is written.

Note: Use this argument with the `-version 3.0` option.

`-delimiter` *`char`*
: Specifies the hierarchy divider character to use in the output file. This option sets the hierarchy divider of only the `write_sdf` command. If omitted, the default hierarchy divider character (`/`) is used as the `write_sdf` default.

`-edges {edged | library | noedge | check_edge}`
: Specifies the edges values. Refer to Table 32-10 on page 1857 for full qualifications of the possible values for the edges option. *Default*: `edged`

  - `check_edge` – Keeps edge specifiers on timing check arcs but does not add edge specifiers on delay arcs.
  - `edged` – Keeps edge specifiers on timing check arcs and delay arcs.
  - `library` – Assumes that the *`sdf_edges`* attributes are specified in the `.lib` file. The default *`sdf_edges`* value of a timing arc is `noedge`.
  - `noedge` – Defined in Table 32-10 on page 1857.

`file_name`
: Specifies the name of the SDF output file. If omitted, this argument defaults to `stdout`.

`-filter`
: Converts all negative `IOPATH` delays to zero.

`-interconn {all | none | noport | noinport | nooutport}`
: Specifies which interconnect delays to write. *Default*: `all`

  - `all` – Writes all `INTERCONN` delays into the SDF file. This is the default if the `-interconn` option is not specified.
  - `none` – No `INTERCONN` delays are written into the SDF file.
  - `noport` – No top level port `INTERCONN` delays are written into the SDF file.
  - `nooutport` – Disables only the output port `INTERCONN` delays.

`-max_view` *`viewName`*
: Uses the late delay from the specified analysis view to populate the SDF max slot, when in multi-mode multi-corner analysis mode.

`-merge_arcs` Merges two arcs that are related to the same internal pin and forms one timing arc. For example, consider a cell's `IOPATH` that forms an arc from pin `I` to internal pin `A` and another from pin `A` to output pin `O`. In this case, if you use the `-merge_arcs` option, the software will merge both the arcs related to internal pin `A` and form one timing arc from pin `I` to output pin `O`.

`-min_period_edges {posedge | negedge | both | none}`

Specifies how to write out minimum `PERIOD` checks. Two check arcs are always created for the `min_period` check when you read in a `.lib` file. The following options let you keep or remove the positive edge, the negative edge, both edges, or no edges from the PERIOD statement in the `.sdf` file.

- `posedge` – Writes out one positive edge `PERIOD` check.
- `negedge` – Writes out one negative edge `PERIOD` check.
- `both` – Writes out both the positive edge and the negative edge `PERIOD` checks. This is the default.
- `none` – Writes out one `PERIOD` check without any edge specifier.

`-min_view` *`viewName`*

Uses the early delay from the specified analysis view to populate the SDF `min` slot, when in multi-mode multi-corner analysis mode.

`-no_condition` Suppresses the `COND` statements in the SDF file.

`-no_escape` Writes out object names without escaping hierarchical dividers. Normally, when writing out an SDF, the hierarchical divider (`/`) is escaped because it is a special character.

`-nonegchecks` Converts all negative timing check values to `0.0`.

`-precision` *`non_neg_integer`*

Controls the number of digits appearing after the decimal point in the output SDF file. The default precision is the value of the `report_precision`.

`-process`

Specifies one or a combination of the following process names that will appear in the header of the generated SDF file:

- `best`
- `typical`
- `worst`

Use the colon delimiter to specify multiple process names. For example:

```
-process best:typical:worst
```

In the above example, the header of the generated SDF file will contain:

```
(PROCESS “best:typical:worst”)
```

`-recrem [merge_always | split | merge_when_paired]`

Determines how RECOVERY and REMOVAL timing checks are written.

- merge_always – Always writes combined RECREM checks, even if either a RECOVERY, or a REMOVAL, check does not exist.

  For example, if a RECOVERY check exists, but a REMOVAL check does not, the software writes a combined RECREM check in the SDF file as follows:

  RECREM (*value*) (*value*) () ()

- split – Splits the RECREM checks into a RECOVERY check and a REMOVAL check.

- merge_when_paired – Writes combined RECREM checks only when both a RECOVERY and a REMOVAL check exist.

  For example, if both a RECOVERY and a REMOVAL check exist, the software writes a combined RECREM check in the SDF file as follows:

  RECREM (*value*) (*value*) (*value*) (*value*)

  However, if a RECOVERY check exists, but a REMOVAL check does not, the software only writes the RECOVERY check in the SDF file:

  RECOVERY (*value*) (*value*)

Default: merge_always

`-remashold`

Converts the SDF v2.1 unsupported REMOVAL (and the REMOVAL portion of the RECREM constructs) to HOLD checks when used with the `-version 2.1` option.
Default: Not to write out the unsupported checks.

Converts the REMOVAL checks to HOLD checks when combined with the `-version 3.0` option.The RECREM checks are split into a RECOVERY check and a HOLD check. This option automatically splits the RECREM checks.
Default: To maintain the version 3.0 checks.

`-resort_values_min_to_max`

Sorts values in the SDF delay triplets into ascending order.

`-scale` *`float`*
Specifies a multiplier to the delay values used in the SDF file. This option does not change the `timescale` setting of the SDF file. The delay values are scaled, but the units are the same.
Default: `1`

`-setuphold [merge_always | split | merge_when_paired]`

Determines how `SETUP and HOLD` timing checks are written.

- `merge_always` – Always writes combined `SETUPHOLD` checks, even if either a `SETUP` or a `HOLD`, check does not exist.

  For example, if a `SETUP` check exists, but a `HOLD` check does not, the software writes a combined `SETUPHOLD` check in the SDF file as follows:

  `SETUPHOLD (`*`value`*`) (`*`value`*`) () ()`

- `split` – Splits the `SETUPHOLD` timing checks into separate `SETUP` and `HOLD` checks.

- `merge_when_paired` – Writes combined `SETUPHOLD` checks only when both a `SETUP` and a `HOLD` check exist.

  For example, if both a `SETUP` and a `HOLD` check exist, the software writes a combined `SETUPHOLD` check in the SDF file as follows:

  `SETUPHOLD (`*`value`*`) (`*`value`*`) (`*`value`*`) (`*`value`*` )`

  However, if a `SETUP` check exists, but a `HOLD` check does not, the software only writes the `SETUP` check in the SDF file:

  `SETUP (`*`value`*`) (`*`value`*`)`

Default: `merge_always`

`-splitrecrem`
Splits the `RECREM` checks into a `RECOVERY` check and a `REMOVAL` check.
Default: To write the combined `RECREM` check.

Note: Use this option only with the `version 3.0` option.

`-splitsetuphold`
Splits the `SETUPHOLD` timing checks into separate `SETUP` and `HOLD` checks.
Default: To write the combined `SETUPHOLD` checks.

`-temperature`

Defines the temperature values that will appear in the header of the generated SDF file.
Use the colon delimiter to specify multiple temperature values. For example:

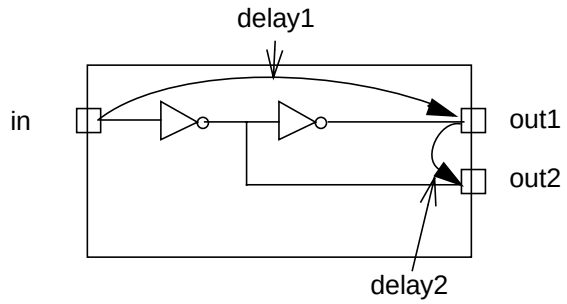
```
-temperature -40.0:25.0:125
```

In the above example, the header of the generated SDF file will contain:

```
(TEMPERATURE -40.0:25.0:125)
```

`-transform_out_to_out_arcs`

Writes out all output-to-output timing arcs as input-to-output arcs. Input-to-output arc delays are computed by adding the delay from the input pin to one of the output pins plus the delay from that output pin to the other output pin. For instance, in the following figure, the delay between `in` and `out2` would be the sum of `delay1` and `delay2`.



`-typ_view` *`viewName`*

Uses the late delay from the specified analysis view to populate the SDF `typ` slot, when in multi-mode multi-corner analysis mode.

`-version {2.1 | 3.0}`

Specifies whether to generate the SDF V2.1 or V3.0. The default SDF version is 3.0. Used in combination with the `-remashold` option.

`-view` *viewName*

Uses the early and late delays for the specified analysis view to populate the `min` and `max` SDF triplet slots, when in multi-mode multi-corner analysis mode. The SDF `typ` slot remains empty (*earlyVal*`::`*lateVal*).

In multi-mode multi-corner analysis mode, if you want to use SPEF information when calculating the delays, you must use `spefIn -rc_corner` to import the SPEF files for each RC corner in the design.

`-voltage`

Defines the voltage values that will appear in the header of the generated SDF file.
Use the colon delimiter to specify multiple voltage values. For example:

```
-voltage 1.980:1.800:1.620
```

In the above example, the header of the generated SDF file will contain:

```
(VOLTAGE 1.980:1.800:1.620)
```

Table 32-10 Definition of -edges Option Values

	edged Value	**library Value**	**noedge Value**
IOPATH from a unate input pin to output pin	No edge specifier	No edge specifier	No edge specifier
IOPATH from a non-unate input pin to output pin	Edge specifier at the input pin	Edge specifier at the input pin when `sdf_edges` is either “`both_edges`” or “`start_edge`”	No edge specifier When merging edged paths, the maximum values are used for MAX and TYP fields; minimum values are used for MIN fields.

Table 32-10 Definition of -edges Option Values, *continued*

	edged Value	library Value	noedge Value
IOPATH of a clock/enable pin to data out pin	Edge specifier at the input pin	Edge specifier at the input pin when `sdf_edges` is either "`both_edges`" or "`start_edge`"	No edge specifier
Timing Checks	Edge specifiers match the check arcs specified in the timing library	Overridden by value of the `sdf_edges` attribute For `both_edges`, the timing checks will contain edge specifier on both starting and ending pins For `start_edge`, the starting pin (for HOLD type check) or ending pin (for SETUP type check) is edge-specified For `end_edge`, the ending pin (for HOLD style check) or the starting pin (for SETUP type check) is edge-specified The timing checks are duplicated or combined appropriately. The `start_edge` and `end_edge` are interpreted as first and second events, respectively, of the Verilog-XL's $hold() and $setup system tasks	Only the edge specifiers of the reference pin are written

Examples

- The following command writes out an SDF 2.1 compliant output file named `my.sdf` with 4 digits after the decimal point:

  ```
  write_sdf -version 2.1 -precision 4 my.sdf
  ```

- The following example SDF files compare the output of `-edges edged` to the output of `-edges noedge`.

  Resultant SDF with `-edges` set to `edged`:

  ```
  (HOLD (posedge D) (posedge CK) (-0.14:-0.14:-0.14))
  (HOLD (negedge D) (posedge CK) (-0.06:-0.06:-0.06))
  ```

```
(SETUP (posedge D) (posedge CK) (0.22:0.22:0.22))
(SETUP (negedge D) (posedge CK) (0.41:0.41:0.41))
```

With `-edges` set to `noedge` only the edge specifiers of the clock pin are written.

```
(HOLD D  (posedge CK) (-0.14:-0.06:-0.06))
(SETUP D  (posedge CK) (0.22:0.41:0.41))
```

write_timing_windows

```
write_timing_windows
     [-sstorm]
     [-pin]
     [-view viewName]
     [-ssta_sigma_multiplier value]
     output_file
```

Generates a Timing Window File (TWF) used by crosstalk analysis tools such as Celtic NDC or delay calculator tools such as SignalStorm™. The TWF file mainly contains the earliest and the latest possible arrival times that a signal may arrive on a net or a pin.

For an example of the Timing Window File format see the *CeltIC NDC Manual*.

Note: SignalStorm requires a pin-based timing windows file. Using the `-sstorm` option, theEncountersoftware writes out pin-based timing windows for SignalStorm.

Parameters

output_file	Writes the TWF file to the specified file name. The *filename* parameter must be the last argument in the list. **Note:** To write a compressed file, add the `.gz` extension to the file name.
`-pin`	Generates a TWF for each pin instead of each net. *Default*: Generates a TWF for the crosstalk analysis tool, which contains timing windows on each net.
`-ssta_sigma_multiplier` *value*	Specifies a sigma multiplier between `0` and `3`. The value that you specify increases the arrival window by that amount and worst slack value is generated. The transition and impedance values are not impacted.
`-sstorm`	Generates a Timing Windows File (TWF) for the SignalStorm delay Calculator tool.
`-view` *viewName*	Generates a TWF for the specified multi-mode multi-corner view only. You must be multi-mode multi-corner analysis mode to use this parameter.

writeDesignTiming

```
writeDesignTiming filename
```

Writes out the timing information of the design.

Use this command when you run the NanoRoute® router in timing-driven mode. The router prioritizes each net to route based on the timing information specified in the output file. Do not edit the file because it is internally generated for the router.

By default, when the router runs with `setNanoRouteMode -routeWithTimingDriven true` specified, it automatically generates a timing information file named `timingfile.tif` in the working directory, and uses it for timing-driven routing.

However, you can control the timing information passed to the router by specifying a timing information file with `writeDesignTiming`. If you specify a file with this command, and also specify `setNanoRouteMode -timingEngine` *`filename`*, the router uses the file written out by `writeDesignTiming` to drive routing, instead of the `timingfile.tif` file.

Parameters

`filename`	Specifies the output filename. **Note:** The output file is for internal tool use only. You must not edit the output file.

writeTimingCon

```
writeTimingCon
     [-dir dirName]
     [-pt]
     [-dc]
     [-filePrefix prefix]
     [-sdc]
```

Generates the timing constraints file in PrimeTime, or Synopsys Design Compiler format.

For a list of constraints written out by the `writeTimingCon` command, see "Constraints Quick Reference," in the *Encounter User Guide*.

Parameters

`-dc`	Generates the constraints file in Synopsys Design Compiler format. **Note:** This parameter is for backward compatibility only. Cadence recommends using the `-pt` parameter to generate the constraints.
`-dir` *`dirName`*	Specifies the name of the directory in which the constraints file will be generated. *Default:* Constraints file is saved in the current directory
`-filePrefix` *`prefix`*	Specifies the filename for the constraints file. *Default:* *`constr`*`.pt`, *`constr`*`.dc`, or *`constr`*`.pks`
`-pt`	Generates the constraints file in PrimeTime format.
`-sdc`	Produces a SDC file where functions to manipulate collections are expanded. For example, the `remove_from_collection all_inputs`, `all_outputs`, `all_clocks`, and filter commands are replaced by the resulting list of objects.

Command Order

Use this command after reading in a timing constraints file.

Example

The following command generates constraints file `newc` in PrimeTime format, and saves the generated file in the directory `new`:

```
writeTimingCon -dir new -pt -filePrefix newc
```

33

Timing Debug Commands

analyze_paths_by_basic_path_group

```
analyze_paths_by_basic_path_group
     [-master master_category_name]
```

Categorizes the timing paths based on basic path groups. The basic path groups categories are:

- Register to register
- Input to register
- Register to output
- Input to output

The following categories are created:

- `REG->REG`
- `IN->REG`
- `REG->OUT`
- `IN->OUT`

Note: If your design does not have paths belonging to a type of path group, the software will not create that category.

For more information on creating path categories, see "Creating Path Categories" in *Encounter User Guide* .

Parameters

`-master` *`master_category_name`*

Specifies that the analysis should be performed only for the specified master category.

analyze_paths_by_bottleneck

```
analyze_paths_by_bottleneck
     -category n
     [-master master_category_name]
```

Creates categories according to the bottleneck analysis of the critical paths. Bottleneck Analysis detects the instances that are often involved in the critical paths in the design. Any change in timing for these instances impacts multiple critical paths in the design. Therefore you can use this information to change your design (floorplan, placement, constraints etc.) around these instances. The bottleneck analysis computes the number of occurrences that an instance appears in the paths that you display in the Timing Debug window, and provides the following output:

- Colors most used instance based on number of occurrences.
- Creates *n* number of path categories for the instances with the highest number of occurrences. *n* is the number you specify using the `-category` parameter.

For more information on creating path categories, see "Creating Path Categories" in *Encounter User Guide* .

Parameters

`-category` *n*
: Specifies the number of categories to be created for most occurring instances.

`-master` *master_category_name*
: Specifies that the analysis should be performed only for the specified master category.

analyze_paths_by_clock_domain

```
analyze_paths_by_clock_domain
     [-include_edges]
     [-master master_category_name]
```

Creates categories according to launch clocks - capture clock combinations.

Following categories are created:

- Paths with clock fully contained in a single domain, clk1.
- Paths with clocks starting one clock domain and ending at another, clk1->clk2.

For more information on creating path categories, see "Creating Path Categories" in *Encounter User Guide* .

Parameters

`-include_edges`	Considers leading or trailing edge of the clock while defining categories.
`-master` *`master_category_name`*	Specifies that the analysis should be performed only for the specified master category.

analyze_paths_by_critical_false_path

```
analyze_paths_by_critical_false_path
     -critical_false_paths
     {false_path_only | exclude_false_path}
     [-master master_category_name]
```

Identifies violating false paths, which are paths that appear as violations but cannot be exercised due to the structure of the netlist, and creates a category that contains them.

For more information on creating path categories, see "Creating Path Categories" in *Encounter User Guide* .

Parameters

`-critical_false_path`

Specifies which paths to analyze.

Default: `false_path_only`

`exclude_false_paths`

Excludes false paths when creating the category.

`false_path_only` Includes only false paths when creating the category.

`-master` *`master_category_name`*

Specifies that the analysis should be performed only for the specified master category.

analyze_paths_by_drv

```
analyze_paths_by_drv
     [[-load_cap_file <capacitance_file>] | [-generate_cap_file <fileName>]]
     [[-load_tran_file <transition_file>] | [-generate_tran_file <fileName>]]
     [-load_fanout_file <fanout_file> | [-generate_fanout_file <fileName>]]
     [-master master_category_name]
```

Reads or generates a report containing max transition, max capacitance, and max fanout violations. Paths that are affected by the selected DRV type are grouped into a category.

For more information on creating path categories, see "Creating Path Categories" in *Encounter User Guide* .

Parameters

`-load_cap_file` *`capacitance_file`*

Reads a file containing a list of paths containing max capacitance violations.

`-load_fanout_file` *`fanout_file`*

Reads a file containing a list of paths containing max fanout violations.

`-load_tran_file` *`transition_file`*

Reads a file containing a list of paths containing max transition violations.

`-generate_cap_file` *`fileName`*

Generates a file containing a list of paths containing max capacitance violations.

`-generate_fanout_file` *`fileName`*

Generates a file containing a list of paths containing max fanout violations.

`-generate_tran_file` *`fileName`*

Generates a file containing a list of paths containing max transition violations.

`-master` *`master_category_name`*

Specifies that the analysis should be performed only for the specified master category.

analyze_paths_by_hierarchy

```
analyze_paths_by_hierarchy
     {{-fp_file floorplan_file | -use_current_floorplan} [-include_macros]}
     | {-partitions partition_list -macros macro_list}
     [-master master_category_name]
```

Creates categories according to the hierarchical characteristics of the paths. The software automatically extracts partition-boundary information from guides, regions, and fences in the floorplan file. Optionally you can specify option to sort paths based on the macros in the floorplan file.

Following categories are created:

- All paths fully contained in a single partition, Ptn1.
- Paths starting in one partition and ending in another partition, Ptn1->Ptn2.
- Paths spanning several partitions, Ptn1->Ptn2->Ptn3.
- Top-level inputs to partitions, INPUT->Ptn1.
- Paths from partitions to top-level outputs, Ptn1->OUTPUT.

For more information on creating path categories, see "Creating Path Categories" in *Encounter User Guide* .

Parameters

`-fp_file floorplan_file`

Specifies the name of the floorplan file.

Default: Uses the current floorplan file.

`-include_macros` Includes macros in addition to guides and fences for creating categories.

Default: Macros are not considered.

`-macros macro_list` Specifies the names of the macros to be used. Use this parameter with the `-partitions` parameter.

`-master master_category_name`

Specifies that the timing paths should be categorized only for the specified master category.

`-partitions partition_list`

Specifies the names of the partitions to consider for extracting hierarchical information.

`-use_current_floorplan`

Uses the current floorplan.

analyze_paths_by_view

```
analyze_paths_by_view
     [-master master_category_name]
```

Categorizes the timing paths based on views that you create during multi-mode multi-corner analysis.

For more information on creating path categories, see "Creating Path Categories" in *Encounter User Guide* .

Parameters

`-master` *`master_category_name`*

Specifies that the timing paths should be categorized only for the specified master category.

create_path_category

```
create_path_category
     -name group_name
     [-overwrite]
     [-sdc {file_name line_number}]
     [-master existing_category_name]
     [-comment {commentText}]
     CONDITIONS
```

Creates path categories using conditions for grouping paths according to analyzed timing results. For more information on analyzing timing results, see "Analyzing Timing Results" in the *Encounter User Guide*.

For more information on timing debug, see the "Debugging Timing Results" in *Encounter User Guide* .

Parameters

`-comment {commentText}`

Specifies a comment that is associated with the category.

CONDITIONS

Defines the conditions that a path must meet to be added to the named category. You can define multiple conditions at a time. The software uses the AND operator to combine all conditions for a complete description of requirements that a path must meet to be added to the category.

A condition can be defined using a combination of various options and types. For a list of options that you can use, see Table 33-1 on page 1876.

Choose from one or more of the following types of conditions:

string

Specifies text to be the condition for a category. For example, a value of BF* would create a category of all paths with names starting with BF.

Float Specifies the comparison operator followed by a floating point number. Choose from the following comparison operators:

==

!=

>

<

>=

<=

For example, you can specify the following:

```
> 1.1
```

```
== 10.1234
```

```
<=- 1.25
```

integer Specifies the comparison operator followed by an integer.

string_plus_float

Specifies the text (string) with comparison operator and a floating-point number. For example, you can specify `BF* == 10.1234`.

string_plus_integer

Specifies the text (string) with comparison operator and an integer.

`-master` *existing_category_name*

Specifies that the new category should be created as a nested category of the specified master category. The master category should be an existing category.

Default: If you do not specify this parameter, the category is created as a new category at the top level.

`-name` *group_name* Specifies the name of the category to be created.

`-overwrite` Overwrites any predefined category with the same name in your category list.

`-sdc {`*file_name line_number*`}`

Creates a category based on the SDC constraint as per the specified SDC file name and line number. Creating a path category based on an SDC constraint helps you to analyze the affect of an SDC constraint by grouping the affected paths together.

Table 33-1 List of Options to Define a Category

`-from_inst | -from_pin | -from_cell | -from_clock | -from_port` *`String`*	Indicates that the path must be from the specified instance, pin, cell, clock, or port pattern.
`-not_from_inst | -not_from_pin | -not_from_cell | -not_from_clock | -not_from_port` *`string`*	Indicates that the path must not be from the specified instance, pin, cell, clock, or port pattern.
`-through_inst | -through_pin | -through_cell | -through_net | -through_port` *`string`*	Indicates that the path must be through the specified instance, pin, cell, net, or port pattern.
`-not_through_inst | -not_through_pin | -not_through_cell | -not_through_net | -not_through_port` *`string`*	Indicates that the path must not be through the specified instance, pin, cell, net, or port pattern.
`-to_inst | -to_pin | -to_cell | -to_clock | -to_port` *`string`*	Indicates that the path must not be to the specified instance, pin, cell, clock, or port pattern.
`-not_to_inst | -not_to_pin | -not_to_cell | -not_to_clock | -not_to_port` *`string`*	Indicates that the path must not be to the specified instance, pin, cell, clock, or port pattern.
`-slack` *`float`*	Indicates that the slack of the path must match the specified *`float`* value.
`-skew` *`float`*	Indicates that the skew of the path must match the specified *`float`* value.
`-launch_latency` *`float`*	Indicates that the launch latency of the path must match the specified *`float`* value.
`-capture_latency` *`float`*	Indicates that the capture latency of the path must match the specified *`float`* value.

`-uncertainty` *`float`*	Indicates that the uncertainty of the path must match the specified *`float`* value.
`-number_of_insts` *`integer`*	Indicates that the number of instances in the data segment (with launch and capture clock paths) must match the specified *`integer`* value.
`-number_of_insts_of_name` *`string_plus_integer`*	Indicates that the number of instances matching the specified instance name match the specified *`integer`* value.
`-number_of_insts_not_of_name` *`string_plus_integer`*	Indicates that the number of instances not matching the specified instance name match the specified *`integer`* value.
`-number_of_insts_of_celltype` *`string_plus_integer`*	Indicates that the number of instances matching the specified celltype match the specified *`integer`* value.
`-number_of_insts_not_of_celltype` *`string_plus_integer`*	Indicates that the number of instances not matching the specified celltype match the specified *`integer`* value.
`-total_delay_of_insts` *`float`*	Indicates that the total delay of instances matches the specified *`float`* value.
`-total_delay_of_path` *`float`*	Indicates that the total delay of path matches the specified *`float`* value.
`-total_delay_of_nets` *`float`*	Indicates that the total delay of nets matches the specified *`float`* value.
`-total_delay_of_insts_of_name` *`string_plus_float`*	Indicates that the total delay of instances matching the specified name matches the specified *`float`* value.
`-total_delay_of_insts_not_of_name` *`string_plus_float`*	Indicates that the total delay of instances not matching the specified name matches the specified *`float`* value.

`-total_delay_of_insts_of_celltype` *`string_plus_float`*	Indicates that the total delay of instances matching the specified celltype matches the specified *`float`* value.
`-total_delay_of_insts_not_of_celltype` *`string_plus_float`*	Indicates that the total delay of instances not matching the specified celltype matches the specified *`float`* value.
`-total_delay_of_nets_of_name` *`string_plus_float`*	Indicates that the total delay of nets matching the specified name matches the specified *`float`* value.
`-total_delay_of_nets_not_of_name` *`string_plus_float`*	Indicates that the total delay of nets not matching the specified name matches the specified *`float`* value.
`-delay_of_any_inst` *`float`*	Indicates that the delay of any instance in the design matches the specified *`float`* value.
`-delay_of_any_net` *`float`*	Indicates that the delay of any net in the design matches the specified *`float`* value.
`-delay_of_every_inst` *`float`*	Indicates that the delay of every instance in the design matches the specified *`float`* value.
`-delay_of_every_net` *`float`*	Indicates that the delay of every instance in the design matches the specified *`float`* value.
`-delay_of_any_inst_of_name` *`string_plus_float`*	Indicates that the delay of any instance in the design matching the specified name matches the specified *`float`* value.
`-delay_of_any_inst_not_of_name` *`string_plus_float`*	Indicates that the delay of any instance in the design not matching the specified name matches the specified *`float`* value.
`-delay_of_any_inst_of_celltype` *`string_plus_float`*	Indicates that the delay of any instance in the design matching the specified celltype matches the specified *`float`* value.

`-delay_of_any_inst_not_of_celltype` *`string_plus_float`*	Indicates that the delay of any instance in the design not matching the specified celltype matches the specified *`float`* value.
`-delay_of_any_net_of_name` *`string_plus_float`*	Indicates that the delay of any net in the design matching the specified name matches the specified *`float`* value.
`-delay_of_any_net_not_of_name` *`string_plus_float`*	Indicates that the delay of any net in the design not matching the specified name matches the specified *`float`* value.
`-delay_of_every_inst_of_name` *`string_plus_float`*	Indicates that the delay of every instance in the design matching the specified name matches the specified *`float`* value.
`-delay_of_every_inst_not_of_name` *`string_plus_float`*	Indicates that the delay of every instance in the design not matching the specified name matches the specified *`float`* value.
`-delay_of_every_inst_of_celltype` *`string_plus_float`*	Indicates that the delay of every instance in the design matching the specified celltype matches the specified *`float`* value.
`-delay_of_every_inst_not_of_celltype` *`string_plus_float`*	Indicates that the delay of every instance in the design not matching the specified celltype matches the specified *`float`* value.
`-delay_of_every_net_of_name` *`string_plus_float`*	Indicates that the delay of every net in the design matching the specified name matches the specified *`float`* value.
`-delay_of_every_net_not_of_name` *`string_plus_float`*	Indicates that the delay of every net in the design not matching the specified name matches the specified *`float`* value.

`-check_type {Setup Check | Hold Check | Clock Gating Setup Check | Library Clock Gating Setup Check | Clock Gating Hold Check | Library Clock Gating Hold Check | External Delay Assertion | Latch Borrowed Time Check | PulseWidth Check | ClockPeriod Check | Recovery Check | Removal Check}`	Indicates that the check type matches the specified option.

delete_path_category

```
delete_path_category
     category_name
```

Deletes the selected category name.

For more information on timing debug categories, see "Debugging Timing Results" in *Encounter User Guide* .

Parameter

`category_name`	Specifies the name of the category to be deleted.

load_path_categories

```
load_path_categories
     -filename filename
```

Loads the path category file in the Timing Debug form. The path category file contains category definitions that you saved in an earlier timing debug analysis. To save path categories, use the `save_path_categories` command.

For more information on timing debug categories, see "Debugging Timing Results" in *Encounter User Guide* .

Parameter

`-filename` *`filename`* Specifies the name of the category file.

load_timing_debug_report

```
load_timing_debug_report
     filename
```

Loads the violation report in Encounter for debugging timing results.

Note: The violation report that you specify in this command has a special machine readable format. Use the `report_timing -machine_readable` command or `timeDesign -timingDebugReport` command to generate the machine readable violation report.

For more information on timing debug categories, see "Debugging Timing Results" in *Encounter User Guide* .

Parameter

filename	Specifies the name of the timing debug file.

save_path_categories

```
save_path_categories
     -filename filename
```

Saves the selected categories in a path category file. You can use the category file to load saved category definitions in a new timing debug session. To load the saved category definitions, use the `load_path_categories` command.

For more information on timing debug categories, see "Debugging Timing Results" in *Encounter User Guide* .

Parameter

`-filename` *`filename`* Specifies the name of the category file.

write_category_summary

```
write_category_summary
     -report fileName
```

Writes a text file containing the following information:

- Category name
- Total number of paths
- Number of passing paths
- Number of failing paths
- Worst negative slack
- Total negative slack

Parameters

`-report` *`fileName`*

Specifies the name of the category report file.

Default: `category.rpt`

write_text_timing_report

```
write_text_timing_report
     [-mtarpt file.mtarpt |
     -category category_name]
     [-path_specification path_specification]
     [-summary] |
     -report file.tarpt
```

Writes a text file containing information about paths that have been loaded from a `.mtarpt` file from Timing Debug, either with the `load_timing_debug_report` command or in the *Display/Generate Timing Report* form in the GUI.

Paths are selected as follows:

- If only one `.mtarpt` file is loaded and no other parameters are specified, then the tool writes all the paths contained in the loaded `.mtarpt` file.
- If there are several `.mtarpt` files loaded, you must specify the `.mtarpt` file from which the tool can select the paths, except if a category is specified.

You can use this report for performing timing analysis.

If only one machine-readable report file is loaded, then the tool writes all paths from that file into the output report. If more than one file is loaded, then the tools issues an error message.

Note: The generated report cannot be read back into the timing debug environment.

Parameters

`-category` *`category_name`*

Specifies the category of paths to write to the text file. You cannot specify `-mtarpt` if you specify this parameter.

`-mtarpt` *`file.mtarpt`*

Specifies the name of the `.mtarpt` file to read and convert to a text file. You cannot specify this parameter if you specify `-category`.

`-path_specification` *`path_specification`*

Writes a single number or path range separated by blank space. A path range is two numbers separated by a minus “-” sign. You can use this parameter only if a single file is specified or loaded.

`-report` *`file.tarpt`*

Specifies the name of the text output file.

`-summary` Writes the file header only. Use this parameter if you do not need a detailed table.

Examples

- The following command loads machine-readable report `top.mtarpt`:

  ```
  load_timing_debug_report top.mtarpt
  ```

- The following command reads the `top.mtarpt` report and reports paths 1 through 50 in the `top_tarpt.txt` file.

  ```
  write_text_timing_report -mtarpt top.mtarpt -path_specification {1-50} -report
  top_tarpt.txt
  ```

34

Timing Budgeting Commands

deriveTimingBudget

```
deriveTimingBudget
     [-setupHold]
     [-freezeTopLevel]
     [-trialIPO | -fullTrialIPO | -noTrialIPO]
     [-constantModel | -noConstantModel]
     [-ptn partitionName | -inst instanceName]
     [-ignoreDontTouch | -noIgnoreDontTouch]
     [-timingCore]
     [-includeLatency | -noIncludeLatency]
     [-ilm]
     [-preliminary]
     [-freezeTopLevelNegPathOnly]
     [-mergeClones instanceName …]
     [-ccd]
     [-justify]
```

Derives timing budgets for the specified hierarchical instances in the design. Use this command after performing floorplanning, placement, and trial routing on the design.

Note: You must free the timing graph to use multiple `deriveTimingBudget` commands with different options.

Encounter operates in multi-mode multi-corner (MMMC) mode, and generates `viewFiles` automatically for partition implementation and top-level chip assembly.

Parameters

`-ccd`	Generates scripts for verification of time budgeting top and block constraints against pre-budget chip constraints using the Conformal Constraint Designer software. For more information, see checkBudgetSdcCCD on page 24.

`-constantModel | -noConstantModel`

Specifies the type of timing model created. The `-constantModel` parameter specifies that constant timing models are created.
The `-noConstantModel` parameter specifies that load-dependent timing models are created.

Default: `-constantModel`

Note: Use the `-noConstantModel` parameter to get a table representation of delays or slews in the partition timing libraries when you perform top-level optimization using timing models.

`-freezeTopLevel` Fixes the top-level timing budget, and proportions the remaining timing budget only for the partitions.

Default: Proportions the timing budget for the top-level and the partitions

`-freezeTopLevelNegPathOnly`

Fixes the top-level timing budget, and proportions the remaining timing budget only for the partitions when the slack is negative.

Default: Proportions the timing budget for the top-level and the partitions

`-includeLatency | -noIncludeLatency`

Specifies whether the clock latency is included in the `set_input_delay` and `set_output_delay` constraints.

Note: The clock latency is always included if the design has propagated clock(s) irrespective of this parameter.

Default: Always includes clock latency.

`-ignoreDontTouch | -noIgnoreDontTouch`

Specifies the handling of don’t touch objects.
The `-noIgnoreDontTouch` parameter handles don’t touch objects as fixed.
The `-ignoreDontTouch` parameter does not handle don’t touch objects as fixed.

Default: `-noIgnoreDontTouch`

`-ilm`

Flattens the design to the ILM view, if not already in that view, and derives budgets only for the ILM partitions.

After budgets are derived, the timing view reverts to the one from which `deriveTimingBudget` was invoked. For example, if `deriveTimingBudget` was invoked from ILM view, it stays in that view, so you can run `justifyBudget` in `flattenIlm` view. If invoked from black box view, `deriveTimingBudget -ilm` returns the design back into black box view. I this case, you cannot run `justifyBudget`.

Note: The `-ilm` parameter is obsolete. This command detects ILMs by default.

`-inst` *`instanceName…`*

Performs budgeting for the hierarchical instance(s). This instance could be a black box. You can derive timing budgets for black boxes and other instances in the same run. In the case of master/clone instances, the software derives budgets for the master partition by default. Specify the `-mergeClones` parameter in addition to the `-inst` parameter to obtain worst-case budgets based on worst-case data per-pin for specified instances in a master/clone set. If you specify instances from more than one master/clone set, the tool derives separate worst-case timing budgets for the specified instances in each set.

Default: If you do not specify this parameter, the software derives budgets for all the partitions.

`-justify`

Justifies the budgets for all ports of the instances or partitions specified by `-inst` or `-ptn`. In MMMC mode, use `deriveTimingBudget -justify` instead `justifyBudget`.

Files are generates per view and per partition. The files are generated with the following structure:

```
budget_justify/Partition1/Partition1_view1.justify
budget_justify/Partition1/Partition1_view2.justify
budget_justify/Partition1/Partition1_view3.justify
budget_justify/Partition2/Partition2_view1.justify
budget_justify/Partition2/Partition2_view2.justify
```

In cases of setup and hold timing budgeting:

```
budget_justify/Partition1/Partition1_view1_setup.justify
budget_justify/Partition1/Partition1_view1_hold.justify
budget_justify/Partition1/Partition1_view2_setup.justify
budget_justify/Partition1/Partition1_view2_hold.justify
```

In non-MMMC mode, files created by `deriveTimingBudget -justifyBudget` are `Hinst_name.justify` or `partition_name.justify`.

`-mergeClones`

Derives a timing budget for hierarchical partitions based on the worst-case data per-pin on the master/clone set containing the partitions. The timing data for the partitions in the master/clone set will be identical, reflecting the worst case among the corresponding pins on the master/clones. You can specify `-mergeClones` regardless of whether you specify `-min`, `-max`, `-critical`, and/or `-combined`. You can use `-mergeClones` with the `-inst` or the `-ptn` parameters. When used with `-inst`, `-mergeClones` derives worst-case budgets for specified instances in a master/clone set. For example, in a master/clone set that includes `instA`, `instB`, and `instC`, if `pin1` has the worst case data on `instA` among pins on `instA`, `instB`, and `instC`, then the software uses the data from `instA/pin1` for `instB/pin1 and instC/pin1`. Similarly, if `pin2` has the worst case data among `instA`, `instB`, and `instC`, then the software uses the data from `instA/pin2` for `instB/pin2` and `instC/pin2`. When used with `-ptn`, `-mergeClones` derives worst-case budgets for all partitions in the master/clone set that contains the specified partition.

Default: The command works without merging the instances or partitions.

`-outfile` *`fileName`*

Directs output to the specified file when you specify the `-justify` parameter.

Default: The software writes reports separately for each hierarchical instance. The reports can be found in the run directory and are named *`HInstance_name`*`.justify` or *`partition_name`*`.justify`.

`-preliminary`

Derives timing budgets in the early stages of the design to get a preliminary estimate of the budgets. When you use this parameter the software uses the net delays and net loads that you specify using the `setThresholdBudgetThreshold` command. The software does not perform calculations to derive the values for net delay and net load, and does not use trial IPO operations for calculating budgets.

The software distributes the positive slack equally between source block, destination block, and top-level. For negative slack, the software uses the `-freezeTopLevelNegPathOnly` parameter.

For more information, see "Deriving Preliminary Budgets in Early Design Phase" in the "Timing Budgeting" chapter of the *Encounter User Guide*.

`-ptn` *`partitionName`*

Performs budgeting for the specified partition(s). Specify the `-mergeClones` parameter in addition to the `-ptn` parameter to obtain worst-case budgets based on worst-case data per-pin for all partitions in the master/clone set containing the partition.

You cannot specify `-ptn` if you use `-inst`.

Default: Performs budgeting on all partitions

`-setupHold`

Derives budgeting for both setup and hold analysis modes. If you do not use this parameter, the software derives budgets for setup mode only. When you use the `setAnalysisMode -checkType hold` command, and do not use the `-setupHold` option, the software still derives budgets for setup mode only, and keeps the mode set to setup for the `justifyBudget` command.

Note: To analyze the derived setup and hold budgets using the `justifyBudget` command, you must set the `setAnalysisMode` command to `-checkType setup` or `-checkType hold`. You can analyze either setup or hold budgets at a time.

`-timingCore`

Generates timing constraints for the partition core (internal register-to-register path only). The following are included in the constraints file:

- Clock definitions
- Core timing constraints such as `set_multicycle_path`, `set_false_path`, and `set_max_delay`.

These constraints are pushed down from the top-level constraints.

Default: Generates timing budget files for each partition from the imported top-level timing constraint file.

Note: The software does not generate budgets for the interface ports of the partition. Therefore the `set_input_delay` or `set_output_delay` values are not generated for the partition ports.

`-trialIPO | -fullTrialIPO | -noTrialIPO`

Inserts trial IPO fixes when generating the timing budget files. These parameters are used only for trial IPO operation, and the Encounter software does not make actual changes in the design.

Default: `-trialIPO`

Choose one of the following:

`-trialIPO`

Runs trial IPO operations on all the top-level nets and the section of the path that is within the partition.

`-fullTrialIPO`

Runs trial IPO operations on all the nets in the design.

`-noTrialIPO` Does not run trial IPO operations.

Examples

- For MMMC examples, see the "Timing Budgeting" chapter of the Encounter User Guide.
- The following examples are for non-MMMC mode only:
  - The following set of commands derives timing budgets for setup and hold analysis and analyzes budgets for setup and hold modes:

```
deriveTimingBudget -ptn -setupHold SH22 SH32
justifyBudget -pins {in in1 out out2} I_SUB
setAnalysisMode -checkType hold
justifyBudget -pins {in in1 out out2} I_SUB
...
saveTimingBudget
```

  - The following commands create a budget for a black box `BBox`:

```
deriveTimingBudget -inst BBox
saveTimingBudget -inst BBox
```

  - The following command derives a worst-case timing budget for instances `A1`, `A2`, and `A3` in master/clone set `{A1, A2, A3, A4, A5}`, and a separate worst-case timing budget for instances `B1`, `B2`, and `B3` in master/clone set `{B1, B2, B3, B4, B5}`:

```
deriveTimingBudget -mergeClones -inst {A1 A2 A3 B1 B2 B3}
```

- The following command derives worst-case timing budgets for all partitions in the master/clone set {A1, A2, A3, A4, A5} containing A1:

  ```
  deriveTimingBudget -mergeClones -ptn A1
  ```

- After you have derived the worst-case timing budget based on -mergeClones, you can justify and save the budgets and save the worst-case timing budgets generated per instance. For examples, see saveTimingBudget on page 1905 and justifyBudget on page 1901.

- The following commands check time budgeted generated top and block constraints under the PARTITION directory against their original pre-budget constraints using the Conformal Constraint Designer (CCD) software. This runs CCD in non-GUI batch mode, and the CCD software exits upon completion of the SDC checking.

  ```
  deriveTimingBudget -ccd
  partition
  savePartition -dir PARTITION
  checkBudgetSdcCCD -partitionDir PARTITION
  ```

Related Topics

- Timing Budgeting chapter of the Encounter User Guide
  - "Deriving Timing Budgets"
    - "Budgeting Using Text Commands"
    - "Top-Level Budgets Derived by Using Virtual Partitioning"
    - "Deriving Preliminary Budgets in Early Design Phase"
  - "Budgeting Output Files for MMMC Designs"
  - "Analyzing Timing Budgets"
    - "Resolving Conflicts with Path-Based Exceptions"
  - "Customizing Budget Generation"
  - "Verifying Timing Budgets"
  - "Reading the Justify Budget Report"

freeTimingBudget

`freeTimingBudget`

Clears budgeting information from memory. Use this command if you do not want to run `saveTimingBudget` or `justifyBudget` after you have run `deriveTimingBudget`.

This command has no parameters.

getBudgetingMode

```
getBudgetingMode
     [-help]
     [-abutted]
     [-localLatency]
     [-localUncertainty]
     [-overrideNetCap]
     [-snapFdBudgetTo]
     [-snapInputBudgetTo]
     [-snapNegativeOnly]
     [-snapOutputBudgetTo]
     [-topLevel]
     [-topLevelDelayPerLen]
     [-topLevelMinDelayPerNet]
     [-quiet]
```

Displays the following information about a specified budgeting mode parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the budgeting mode parameters.

Parameters

`-parameter_names`

Displays information for the specified parameters. You can specify one or more parameters.

See `setBudgetingMode` for descriptions of the budgeting mode parameters you can specify.

`-quiet`

Displays the current settings for the specified parameters in Tcl list format only.

If you specify `-quiet` without any parameters, the software displays the current settings of all `setBudgetingMode` parameters in Tcl list format.

Examples

- The following command displays the current setting for the `-abutted` parameter:

  ```
  getBudgetingMode -abutted
  ```

  The software displays the following information:

  ```
  -abutted false                  # bool, default=false
  false
  ```

- The following command displays the current setting for the `-abutted` parameter in Tcl list format:

  ```
  getBudgetingMode -abutted -quiet
  ```

  The software displays the following information:

  ```
  false
  ```

- The following command displays the current settings for the `-localLatency` and `-localUncertainty` parameters:

  ```
  getBudgetingMode -localLatency -localUncertainty
  ```

  The software displays the following information:

  ```
  -localLatency false             # bool, default=false
  -localUncertainty false         # bool, default=false
  {localLatency false} {localUncertainty false}
  ```

justifyBudget

```
justifyBudget
     -pins pinList
     [-outfile fileName]
     [-short]
     {partitionName | instanceName}
     [-mergeClones]
     [-view viewName]
```

Queries budgets for specified partitions or instance ports and generates a report. You can query how budgets were generated for black box ports.

Use this command after partitioning the design using the `deriveTimingBudget` command, and after saving the generated budgets.

Note: In case of combined budgeting, the `justifyBudget` command will justify all the input or output delays attached to the port of a partition.

Important

The `justifyBudget` command only supports the `-short` parameter in MMMC mode (the default mode). The report will contain only the headers. To print paths, use `deriveTimingBudget -justify`.

Parameters

instanceName	Specifies the name of the instance to analyze. When used with `-mergeClones`, the software reports the worst-case data for specified instance pins in the master/clone set containing the instances.

`-mergeClones`	Generates a debug report for each pin on each master/clone based on the worst timing data among the corresponding pins on all the master/clones. You must first derive the timing budget using `deriveTimingBudget -mergeClones`. Given a negative slack, this parameter distributes the total available timing between port of the instances along a path based on worst-case pins. The command behaves differently, depending on whether you specify an instance or partition. If you derived the timing budget for a master/clone partition, the software reports data for all pins on all partitions in the master/clone set if you specify `-ptn` along with this `-mergeClones` parameter. If you derived the timing budget for specified instances in a master/clone set, the software reports data for the specified instances if you specify `-inst` along with this `-mergeClones` parameter. See the examples below.
`-outfile` *`fileName`*	Specifies the output file name that contains timing budget data for a pin.
`partitionName`	Specifies the name of the partition to analyze. When used with `-mergeClones`, the software reports the worst-case data for specified partitions pins in the master/clone set containing the partition.
`-pins` *`pinList`*	Specifies the hierarchical partition pin or pins names to be analyzed. You can supply one or more pin names, or `*` to specify all pins.
`-short`	Prints the header information of the `justifyBudget` report. When you run `deriveTimingBudget` in MMMC mode, `justifyBudget` runs only with this parameter.
`-view` *`viewName`*	Prints a report for the specified view. If you derived the timing budget in MMMC mode (default), you must use this parameter to obtain a report.

Examples

- For examples of budgeting for MMMC designs, see the “Timing Budgeting” chapter of the *Encounter User Guide*.
- The following command prints a report for view `VIEW1`

```
justifyBudget -view VIEW1
```

- Prints header information for budgets created by `deriveTimingBudget` in MMMC mode (default):

  ```
  justifyBudget -short
  ```

The following examples do not apply in MMMC mode: they are shown only for backward compatibility.

- The following command outputs the timing budget data for pin `MCB<0>`:

  ```
  justifyBudget -pin MCB<O>
  ```

- The following commands outputs the timing budget data for a list of pins in `I_SUB`:

  ```
  justifyBudget -pins in I_SUB
  justifyBudget -pins {in in1 out out2} I_SUB.
  ```

- The following commands derives the timing budgets, justifies the budgets for both setup and hold analysis, and saves the budgeting results:

  ```
  deriveTimingBudget -ptn partitionName -combined -setupHold
  justifyBudget -pin in1 partitionName
  setAnalysisMode -checkType hold
  justifyBudget -pin in1 partitionName
  saveTimingBudget -ptn partitionName -dir tmp
  ```

- The following command derives the worst-case timing budget for all master/clones in the master/clone set containing partition `LEFT`.

  ```
  deriveTimingBudget -mergeClones -ptn LEFT
  ```

  You must run this command before you justify the budget.

  The following command generates a report for `pin1` on `LEFT`, where the timing data on `pin1` reflects the worst case timing among the corresponding pins on all partitions in the master/clone set that includes `LEFT`.

  ```
  justifyBudget -mergeClones -pins pin1 LEFT
  ```

- The following command derives the worst-case timing budget for instance `LEFT` and `RIGHT` in the master/clone set containing `LEFT` and `RIGHT`:

  ```
  deriveTimingBudget -mergeClones -inst LEFT RIGHT
  ```

  You must run this command before you justify the budget.

- The following command generates a report for `pin1` on `LEFT`, where the timing data on `pin1` reflects the worst case timing among the corresponding pin on the instances (`LEFT` and `RIGHT`) in the master/clone set that includes `LEFT and RIGHT:`

  ```
  justifyBudget -mergeClones -pins pin1 LEFT
  ```

Related Topics

- Timing Budgeting chapter of the Encounter User Guide
  - "Deriving Timing Budgets"
    - "Top-Level Budgets Derived by Using Virtual Partitioning"
    - "Deriving Preliminary Budgets in Early Design Phase"
  - "Budgeting Output Files for MMMC Designs"
  - "Verifying Timing Budgets"
  - "Reading the Justify Budget Report"
  - "Generated Report for Design Example"

saveTimingBudget

```
saveTimingBudget
     [-dir dirName]
     [-pt]
     [-lib]
     [-snapBudget]
     [-pinLoad]
     [-driveCell | -inputTransition]
     [-mergeClones]
     [-latencyOnClocks]
     [-rptNegSlackOnPorts value]
     [-noFalsePathsForUnCstrPorts]
     [-ptn {pinList} | -inst {instList}]
     [-writeFPForHold]
     [{ instList }]
```

Saves time budget files of specified hierarchical instances to a specified directory. This command also saves the Stamp model files.

Use this command after deriving the time budgets using the `deriveTimingBudget` command.

Parameters

`-dir` *`dirName`*	Specifies the directory name that contains the derived timing budget files. The files are derived in a subdirectory *`dirName/cellTypeName/cellTypeName`*`.constr.pt`. If you do not specify this parameter, the timing budget files are derived in the current directory.
`-driveCell`	Writes out the boundary drive information using the `set_driving_cell` constraint.
`-inputTransition`	Writes out input transition time in the budgeting output files using the `set_input_transition` constraint.
`instList`	Saves the budget files for instances. You use this parameter with the `deriveTimingBudget -inst` command.
`-latencyOnClocks`	Writes the `set_clock_latency` constraint on the clocks instead of ports in the budgeting files. Use this parameter when you have multiple clocks on the same port. *Default:* Writes the `set_clock_latency` constraint on the ports.
`-lib`	Saves the library model.

`-mergeClones` Saves the timing budgets for the masters/clones based on the worst-case timing data per pin. When used with the `-inst` parameter, `-mergeClones` saves the budget based on the worst-case data for the specified instances in the master/clone set. When used with `-ptn`, `-mergeClones` saves the budget based on the worst-case data for all the partitions in the master/clone set.

`-noFalsePathsForUnCstrPorts`

Specifies not to create false paths for unconstrained ports.

`-pinLoad` Writes out the `setload` with `-pin_load` information.

`-pt` Specifies PrimeTime format.

`-ptn {`*`ptnList`*`} | -inst {`*`instList`*`}`

Saves the budget file for partitions or instances for which you have generated timing budgets. You do not need to specify `-ptn` or `-inst` if you have previously specified `deriveTimingBudget -ilm`.

The `-ptn` parameter saves the budget files for partitions. You use this parameter with the `deriveTimingBudget -ptn` command.

The `-inst` parameter saves the budget files for instances. You use this parameter with the `deriveTimingBudget -inst` command.

Default: The tool saves budgets for all instances and partitions specified by `deriveTimingBudget`.

`-rptNegSlackOnPorts` *`value`*

Reports if partition ports have a slack less than the specified value and saves this information in the `.warn` file. The slack is calculated based on data arrival time and clock time at data sampling points. The report is saved to the `.warn` file.

`-snapBudget` Assigns a percent of the external delay value to the module or instance name when the overall timing budget is exceeded. The external delay is reduced by the specified percentage in this case. If you do not specify this parameter, the software does not exceed the timing budget.

You specify the percentage to be used for this parameter using the `setBudgetingMode` command.

`-writeFPForHold`

Writes `false_path -hold` constraints for all I/Os when deriving timing budgets in setup mode (the default behavior when you do not specify the `-setupHold` parameter).

For inputs, the `saveTimingBudget` command writes the following constraint:

```
set_false_path -from port -hold
```

For outputs, the `saveTimingBudget` command writes the following constraint:

```
set_false_path -to port -hold
```

Examples

- The following command saves a timing budget file in SDC format for `SH17` in a directory `try3`:

  ```
  saveTimingBudget -dir try3 sheet17
  ```

- The following commands save timing budgets for specified master/clone instances.

  First, derive the timing budgets for the `LEFT` and `RIGHT` instances, based on the worst-case timing data perm pin:

  ```
  derviveTimingBudget -mergeClones -inst LEFT RIGHT
  ```

  Save the timing budget for instance `LEFT` in directory `LCLONE`:

  ```
  saveTimingBudget -dir LCLONE -pt -inst LEFT
  ```

  Save the timing budget for instance `RIGHT` in directory `RCLONE`:

  ```
  saveTimingBudget -dir RCLONE -pt -inst RIGHT
  ```

  Save the worst-case timing budget based on `LEFT` and `RIGHT` clones:

  ```
  saveTimingBudget -dir WostCase -mergeClones -inst LEFT RIGHT
  ```

- The following commands save timing budgets for master/clone partitions when worst-case budgets were derived for all partitions in a master/clone set. Partitions `LEFT` and `RIGHT` are contained in the master/clone set.

  First, derive the worst-case timing budget for all partitions in the master/clone set that includes partition `RIGHT`:

  ```
  deriveTimingBudget -mergeClones -ptn RIGHT
  ```

  You do not need to specify partition `LEFT`. With the `-ptn` option, this command derives budgets based on worst-case data per pin for all partitions in the master/clone set, regardless of which partition you specify.

Save the timing budget in directory `RCLONE` for partition `RIGHT` based on worst-case data per pin from all partitions.

```
saveTimingBudget -dir RCLONE -pt -mergeClones -ptn RIGHT
```

Save the timing budget in the directory `LCLONE` for partition `LEFT` based on worst-case data per pin from all partitions:

```
saveTimingBudget -dir LCLONE -pt -mergeClones -ptn LEFT
```

In fact, you can specify any partition in the master/clone set, for example:

```
saveTimingBudget -dir LCLONE -pt -mergeClones -ptn RIGHT
```

Related Topics

- Timing Budgeting chapter of the Encounter User Guide
  - "Deriving Timing Budgets"
    - "Budgeting Using Text Commands"
    - "Top-Level Budgets Derived by Using Virtual Partitioning"
    - "Deriving Preliminary Budgets in Early Design Phase"
  - "Analyzing Timing Budgets"
  - "Customizing Budget Generation"
  - "Verifying Timing Budgets"
  - "Reading the Justify Budget Report"
  - "Warning Report"
    - "Pin Constraint Values Greater than Available Time"

setBudgetingMode

```
setBudgetingMode
     [-help]
     [-reset]
     [-abutted {true|false}]
     [-localLatency {true|false}]
     [-localUncertainty {true|false}]
     [-overrideNetCap value]
     [-snapFdBudgetTo value]
     [-snapInputBudgetTo value]
     [-snapNegativeOnly {true|false}]
     [-snapOutputBudgetTo value]
     [-topLevel value]
     [-topLevelDelayPerLen value]
     [-topLevelMinDelayPerNet value]
```

Sets the timing budgeting mode.

Parameters

`-abutted {true | false}`

Specifies that the software abuts the partitions when allocating budgets from positive slack paths. With this parameter, the software assumes that all partitions are abutted and therefore the software does not allocate extra delay at the top-level.

Note: The software automatically handles the abutted partitions (both partial and full) even when you do not specify this parameter.

`-help`

Outputs a brief description that includes type and default information for each `setBudgetingMode` parameter. For a detailed description of the command and all of its parameters, use the man command: man `setBudgetingMode`.

`-localLatency {true | false}`

Adds the `set_clock_latency` constraint information that you specify using the `setPtnUserCnsFile` command to the constraints file generated after budgeting.

When you specify this parameter, the software checks whether the `set_clock_latency` constraint specifies a clock or a port.

If the constraint specifies a clock and the clock exists in the partition, then the constraint is added to the constraints file generated after budgeting. If the clock does not exist in the partition, the software does not write out the constraint.

If the constraint specifies a port, the constraint is added to the constraints file generated after budgeting.

The source latency is calculated as follows:

source latency = top-level source latency + top-level network latency - local network latency

`-localUncertainty {true | false}`

Adds the `set_clock_uncertainty` constraint information that you specify using the `setPtnUserCnsFile` command to the constraints file generated after budgeting.

When you use this parameter, the constraint file generated after budgeting has input and output delays using virtual clocks instead of real clocks in the SDC file even if you use the `-min`, `-max`, or `-critical` parameters in the `deriveTimingBudget` command. The software uses the virtual clocks to differentiate the clock uncertainty for external paths (paths that are coming from or going to the top-level) and internal paths of the partition.

`-overrideNetCap` *`value`*

Specifies the lump capacitance value. The software uses this value when you use the `-preliminary` parameter in the `deriveTimingBudget` command.

Default: `100ff`

`-reset`

Resets parameters to their default values. The `-reset` parameter must be the first parameter specified. If you specify `-reset` by itself, the software resets all `setBudgetingMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-snapFdBudgetTo` *`value`*

Specifies the minimum delay value (in picoseconds) for the path from partition input port to partition output port. If, during timing analysis, the path segment delay used to generate budgeting constraints for the partition ports falls below the specified threshold value, the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for the path.

The `.warn` file does not issue a warning for feedthrough segments. By default, the software distributes a delay equivalent of two medium-sized buffer delays among the feedthrough segments. As a result, it is unlikely that the feedthrough segment delay will fall below a reasonable value.

`-snapInputBudgetTo` *`value`*

Specifies the minimum delay (in picoseconds) for the path from the partition input port to the internal register. If, during timing analysis, the path segment delay falls below the specified threshold value, the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for the path.

Note: During timing analysis, the path segment delay could be greater then the specified value; however, after budget allocations, the allocated delay for the partition can still become smaller than the specified threshold value. The software checks the delays allocated to partitions for input and output paths and creates appropriate warning messages in the partition `.warn` file.

`-snapNegativeOnly`

Considers the only negative slack paths when used with `-snapInputBudgetTo` and `-snapOutputBudgetTo`. By default, `-snapInputBudgetTo` and `-snapOutputBudgetTo` consider both positive and negative slack.

`-snapOutputBudgetTo` *`value`*

Specifies the min delay (in picoseconds) for the path from the internal register to the partition output port. If, during timing analysis, the path segment falls below the specified threshold value, then the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for the path.

`-topLevel` *`minimumValue`*

Specifies the minimum percentage of total available budget set aside for the top level.

Default: `0`

`-topLevelDelayPerLen` *`value`*

Specifies the top-level estimated delay (in picoseconds) per millimeter length. The software uses this value when you use the `-preliminary` parameter in the `deriveTimingBudget` command.

For more information on the use of the `-topLevelDelayPerLen` parameter, see "Deriving Preliminary Budgets in Early Design Phase" in the "Timing Budgeting" chapter of the *Encounter User Guide*.

Default: `180ps/mm`

`-topLevelMinDelayPerNet` *`value`*

Specifies the top-level minimum delay-per-net value in picoseconds/millimeter units. The software uses this value when you use the `-preliminary` parameter in the `deriveTimingBudget` command.

For more information on the use of the `-topLevelMinDelayPerNet` parameter, see "Deriving Preliminary Budgets in Early Design Phase" in the "Timing Budgeting" chapter of the *Encounter User Guide*.

Default: `100ps/mm`

Examples

The following command sets the input and output budgets to 3000.

```
setBudgetingMode -snapInputBudget 3000 -snapOutputBudget 3000
```

The resulting message appear in the `.warn` file when you generate budgets.

```
/* Start Section: Pin constraint values less than defined threshold time
(setup)*/
Input Threshold Value:   3.000
Output Threshold Value:  3.000
Constraint: set_output_delay 11.524 -clock {CLK2} -max -rise [get_ports
{sub1_out1}]
allocates block delay less than input  threshold time.
Block delay: 0.476

Constraint: set_output_delay 11.524 -clock {CLK2} -max -fall [get_ports
{sub1_out1}]
allocates block delay less than input  threshold time.
Block delay: 0.476
/* End Section: Pin constraint values less than defined threshold time (setup)*/
```

`justifyBudget` header example:

```
Budgeted constraint type: set_output_delay(setup rise)

Initial budget available time = 12.000

One Buffer Delay for Adjustment(cell ssiv): 1.000
Fixed Delay for Feedthrough Paths(fixFdThru)= (2 x 1.000) = 2.000
External Segment Fixed Delay From Budget Snap(snapExtFixedDel) = 5.000
Total External Segment Fixed Delay(extFixDel) = 5.000
This Block's Segment Fixed Delay from budget snap(snapIntFixedDel) = 1.000
Total This Block's Segment Fixed Delay(intFixDel) = 1.000
External Segment Extra Delay From Budget Snap (snapExtDelExtra) = 2.000
This Block's  Extra Delay From Budget Snap (snapIntDelExtra) = 1.000
Path Extra Delay From Budget Snap (snapExtraDel)  = (2.000 + 1.000) = 3.000
Fixed Delay on the Path(pathFixDel) = (5.000 + 1.000 + 1.000 + 2.000) = 9.000
Fixed Delay Adjustment(fixDel)= 9.000
Clock Skew(clkSkew): 0.000
Adjustment for budget available time= -(pathFixDel + fixFdThru - clkSkew +
snapExtraDel)

= -(6.000 + 2.000 - 0.000 + 3.000)

= -11.000

Available budget after adjustments(AvailTime)= (12.000 - 11.000) = 1.000

External Segment Delay(extSegDel): 8.000
This Block's Segment Delay(segDel): 0.000
Total delay(totDel): 8.000 + 0.000 = 8.000
Virtual Clock Adjustment(virClk): 0.000
Budgeted constraint = AvailTime * extSegDel / totDel + fixDel + virClk + startClkLat
Budgeted constraint = 1.000 * 8.000 / 8.000 + 9.000 + 0.000 + 0.000 = 10.000
```

Related Topics

- Timing Budgeting chapter of the Encounter User Guide

setPtnUserCnsFile

```
setPtnUserCnsFile
     -fileName name
     -ptnName partitionName
     [-view]
```

Specifies the constraints file for a specific partition. The software appends the constraint information in this file to the timing constraints generated when you use the savePartition or saveTimingBudget command. You can specify the following constraints in the constraints file for the partition:

- set_clock_uncertainty
- set_clock_latency (network latency constraints)

You must specify the -localLatency and -localUncertainty parameters in the setBudgetingMode command before using the setPtnUserCnsFile command.

Parameters

-fileName *name*

Specifies the name of the file containing constraints specific to a partition.

-ptnName *partitionName*

Specifies the name of the existing partition.

-view

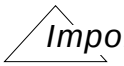
Specifies a partition-specific constraint file for each MMMC view.

setThresholdBudgetRatio

```
setThresholdBudgetRatio
     [-topLevel minimumValue]
     [-abutted]
     [-localUncertainty]
     [-localLatency]
     [-overrideNetCap value]
     [-snapFdBudgetTo value]
     [-snapInputBudgetTo value]
     [-snapOutputBudgetTo value]
     [-snapNegativeOnly]
     [-topLevelDelayPerLen value]
     [-topLevelMinDelayPerNet value]
```

Sets the threshold budget ratio.

Use this command before running timing budgeting commands such as `deriveTimingBudget`.



Important

Command `setThresholdBudgetRatio` is obsolete and has been replaced by the `setBudgetingMode` command. All the parameters of this command are now available with the `setBudgetingMode` command.

Parameters

`-abutted`	Specifies that the software abuts the partitions when allocating budgets from positive slack paths. With this parameter, the software assumes that all partitions are abutted and therefore the software does not allocate extra delay at the top-level. **Note:** The software automatically handles the abutted partitions (both partial and full) even when you do not specify this parameter.

`-localLatency`

Adds the `set_clock_latency` constraint information that you specify using the `setPtnUserCnsFile` command to the constraints file generated after budgeting.

When you specify this parameter, the software checks whether the `set_clock_latency` constraint specifies a clock or a port.

If the constraint specifies a clock and the clock exists in the partition, then the constraint is added to the constraints file generated after budgeting. If the clock does not exist in the partition, the software does not write out the constraint.

If the constraint specifies a port, the constraint is added to the constraints file generated after budgeting.

The source latency is calculated as follows:

source latency = top-level source latency + top-level network latency - local network latency

`-localUncertainty`

Adds the `set_clock_uncertainty` constraint information that you specify using the `setPtnUserCnsFile` command to the constraints file generated after budgeting.

When you use this parameter, the constraint file generated after budgeting has input and output delays using virtual clocks instead of real clocks in the SDC file even if you use the `-min`, `-max`, or `-critical` parameters in the `deriveTimingBudget` command. The software uses the virtual clocks to differentiate the clock uncertainty for external paths (paths that are coming from or going to the top-level) and internal paths of the partition.

`-overrideNetCap` *`value`*

Specifies the lump capacitance value. The software uses this value when you use the `-preliminary` parameter in the `deriveTimingBudget` command.

Default: `100ff`

`-snapFdBudgetTo` *`value`*

Specifies the minimum delay value (in picoseconds) for the path from partition input port to partition output port. If, during timing analysis, the path segment delay used to generate budgeting constraints for the partition ports falls below the specified threshold value, the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for the path.

The `.warn` file does not issue a warning for feedthrough segments. By default, the software distributes a delay equivalent of two medium-sized buffer delays among the feedthrough segments. As a result, it is unlikely that the feedthrough segment delay will fall below a reasonable value.

`-snapInputBudgetTo` *`value`*

Specifies the minimum delay (in picoseconds) for the path from the partition input port to the internal register. If, during timing analysis, the path segment delay falls below the specified threshold value, the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for the path.

Note: During timing analysis, the path segment delay could be greater then the specified value; however, after budget allocations, the allocated delay for the partition can still become smaller than the specified threshold value. The software checks the delays allocated to partitions for input and output paths and creates appropriate warning messages in the partition `.warn` file.

`-snapNegativeOnly`

Considers the only negative slack paths when used with `-snapInputBudgetTo` and `-snapOutputBudgetTo`. By default, `-snapInputBudgetTo` and `-snapOutputBudgetTo` consider both positive and negative slack.

`-snapOutputBudgetTo` *`value`*

Specifies the min delay (in picoseconds) for the path from the internal register to the partition output port. If, during timing analysis, the path segment falls below the specified threshold value, then the delay segment is snapped to the specified value and is considered as a fixed delay during budget allocation for the path.

`-topLevel` *`minimumValue`*

Specifies the minimum percentage of total available budget set aside for the top level.

Default: `0`

`-topLevelDelayPerLen` *`value`*

Specifies the top-level estimated delay (in picoseconds) per millimeter length. The software uses this value when you use the `-preliminary` parameter in the `deriveTimingBudget` command.

For more information on the use of the `-topLevelDelayPerLen` parameter, see "Deriving Preliminary Budgets in Early Design Phase" in the "Timing Budgeting" chapter of the *Encounter User Guide*.

Default: `180ps/mm`

`-topLevelMinDelayPerNet` *`value`*

Specifies the top-level minimum delay-per-net value in picoseconds/millimeter units. The software uses this value when you use the `-preliminary` parameter in the `deriveTimingBudget` command.

For more information on the use of the `-topLevelMinDelayPerNet` parameter, see "Deriving Preliminary Budgets in Early Design Phase" in the "Timing Budgeting" chapter of the *Encounter User Guide*.

Default: `100ps/mm`

Examples

The following command sets the input and output budgets to 3000.

```
-setThresholdBudgetTo -snapInputBudget 3000 -snapOutputBudget 3000
```

The resulting message appear in the `.warn` file when you generate budgets.

```
/* Start Section: Pin constraint values less than defined threshold time
(setup)*/
Input Threshold Value:   3.000
Output Threshold Value:  3.000
Constraint: set_output_delay 11.524 -clock {CLK2} -max -rise [get_ports
{sub1_out1}]
allocates block delay less than input  threshold time.
Block delay: 0.476

Constraint: set_output_delay 11.524 -clock {CLK2} -max -fall [get_ports
{sub1_out1}]
allocates block delay less than input  threshold time.
Block delay: 0.476
/* End Section: Pin constraint values less than defined threshold time (setup)*/
```

`justifyBudget` header example:

```
Budgeted constraint type: set_output_delay(setup rise)

Initial budget available time = 12.000

One Buffer Delay for Adjustment(cell ssiv): 1.000
Fixed Delay for Feedthrough Paths(fixFdThru)= (2 x 1.000) = 2.000
External Segment Fixed Delay From Budget Snap(snapExtFixedDel) = 5.000
Total External Segment Fixed Delay(extFixDel) = 5.000
This Block’s Segment Fixed Delay from budget snap(snapIntFixedDel) = 1.000
Total This Block’s Segment Fixed Delay(intFixDel) = 1.000
External Segment Extra Delay From Budget Snap (snapExtDelExtra) = 2.000
This Block’s  Extra Delay From Budget Snap (snapIntDelExtra) = 1.000
Path Extra Delay From Budget Snap (snapExtraDel)  = (2.000 + 1.000) = 3.000
Fixed Delay on the Path(pathFixDel) = (5.000 + 1.000 + 1.000 + 2.000) = 9.000
Fixed Delay Adjustment(fixDel)= 9.000
Clock Skew(clkSkew): 0.000
Adjustment for budget available time= -(pathFixDel + fixFdThru - clkSkew +
snapExtraDel)

= -(6.000 + 2.000 - 0.000 + 3.000)

= -11.000

Available budget after adjustments(AvailTime)= (12.000 - 11.000) = 1.000

External Segment Delay(extSegDel): 8.000
This Block’s Segment Delay(segDel): 0.000
Total delay(totDel): 8.000 + 0.000 = 8.000
Virtual Clock Adjustment(virClk): 0.000
Budgeted constraint = AvailTime * extSegDel / totDel + fixDel + virClk + startClkLat
Budgeted constraint = 1.000 * 8.000 / 8.000 + 9.000 + 0.000 + 0.000 = 10.000
```

Related Topics

- Timing Budgeting chapter of the Encounter User Guide
  - “Deriving Timing Budgets”
    - “Deriving Preliminary Budgets in Early Design Phase”
  - “Budgeting Output Files for MMMC Designs”

35

Timing Global Variables

Timing Global Commands

The following commands retrieve and set the timing global variable values:

get_global

```
get_global
     global_variable
```

Retrieves the current value of the specified timing global variable. To set the value of a timing global variable, use the `set global` command.

For a list of timing global variables, see Timing Global Variables on page 1927.

Example

The following command retrieves the setting for the `timing_cppr_threshold_ps` global.

```
get_global timing_cppr_threshold_ps
```

report_globals

```
report_globals
     [pattern]
     [-add addtional_timing_globals]
```

Generates a list of the officially supported timing global variables, with their current and default values.

Parameters

`-add` *`additional_timing_globals`*

Adds the specified Beta or customer-specific timing global variables to the list of global variables that are reported. When you use the `report_globals` command, these global variables will then be included in the generated list, with their registered and current values.

Note: This parameter only adds the specified global variables to the variable list. It does not generate the list itself.

`pattern`

Reports all timing global variables that match the specified pattern. Pattern matching can be used for both application globals, and customer-specified globals (that is, globals registered with the `report_globals -add` parameter).

set_global

```
set_global
     global_name
     value
```

Sets the specified timing global variable to a specified value. To retrieve the current value of a timing global variable, use the `get_global` command.

For a list of timing global variables, see Timing Global Variables on page 1927.

Example

The following command sets the `timing_cppr_threshold_ps` global.

```
set_global timing_cppr_threshold_ps 25
```

Timing Global Variables

The following timing global variables can be set and queried with the `set_global` and `get_global` commands:

To set this global variable, use the `set global` command.

case_analysis_sequential_propagation

```
case_analysis_sequential_propagation
     {true | false}
```

Default: `false`

Important

Global variable “`case_analysis_sequential_propagation`” is obsolete and has been replaced by “`timing case analysis for sequential propagation`”.

clock_gating_to_be_checked

```
clock_gating_to_be_checked
     {true | false}
```

Default: `true`

Important

Global variable “`clock_gating_to_be_checked`” is obsolete and has been replaced by “`timing_disable_clock_gating_checks`”.

lib_build_asynch_arc

```
lib_build_asynch_arc
     {true | false}
```

Default: `false`

Important

Global variable "`lib_build_asynch_arc`" is obsolete and has been replaced by "`timing_enable_preset_clear_arcs`".

lib_build_asynch_de_assert_arc

```
lib_build_asynch_de_assert_arc
     {true | false}
```

Default: `true`

Controls whether input to output arcs from the preset or clear pins transitioning to inactive state are included when the timing system is initialized.

When the control signal to the preset or clear pins transitions to inactive state (deassertion), in some cases, the output pin can subsequently transition. In the case of edge-triggered flip-flops, the `removal` timing check (a type of hold check) prevents the preset or clear from deasserting too close to the active edge of the clock; therefore, there is usually no chance of the output changing, and no need to have an additional arc to model this transition.

However, in the case of latches, it is possible that the `removal` constraint will still allow the signal to deassert when the latch is in transparent mode. If the input state of the latch is not equivalent to the state previously forced by the preset or clear, an output transition results. In these situations, an additional arc is required to model this transition. A similar corner-case can result when both preset and clear signals are asserted, and one of them is subsequently released.

Note: If both the `timing_enable_preset_clear_arcs` and the `lib_build_asynch_de_assert_arc` global variables are set to `true`, all asynchronous arcs are recognized. However, if `timing_enable_preset_clear_arcs` is set to `true` and `lib_build_asynch_de_assert_arc` is set to `false`, only assert asynchronous arcs are recognized.

Because the `timing_enable_preset_clear_arcs` and `lib_build_asynch_de_assert_arc` global variables affect how the timing graph is built, it is important to set the desired state before performing any timing analysis.

To set this global variable, use the `set_global` command.

lib_build_timing_cond_default_arc

```
lib_build_timing_cond_default_arc
     {true | false}
```

Default: `true`

Controls the creation of the default timing arc. When set to `true`, the default timing arc is created from the conditional timing arc delays specified in the library. The created default timing arc does not have the WHEN condition. When set to `false`, the default timing arc from the library is disabled.

To set this global variable, use the `set global` command.

locv_chip_size

`locv_chip_size` *`real_number`*

Default: Automatically calculated from SPEF file, if one is loaded.

Specifies the diagonal length of the chip, in microns.

The chip and core sizes specified with the `locv_chip_size` and `locv_core_size` global variables take priority over chip and core size values derived from the SPEF file, or through placement. In the case that there is no SPEF file loaded, or placement information available:

- If you specify `locv_chip_size` but not `locv_core_size`, the software considers the core size to be equal to the `locv_chip_size` value.
- If you specify `locv_core_size` but not `locv_chip_size`, the software ends the LOCV analysis and generates an error message.
- If both `locv_chip_size` and `locv_core_size` are not specified, the software ends the LOCV analysis and generates an error message.

To set this global variable, use the `set global` command.

locv_core_size

`locv_core_size` *`real_number`*

Default: Automatically calculated from SPEF file, if one is loaded.

Specifies the diagonal length of the core area, in microns.

The chip and core sizes specified with the `locv_chip_size` and `locv_core_size` global variables take priority over chip and core size values derived from the SPEF file, or through placement. In the case that there is no SPEF file loaded, or placement information available:

- If you specify `locv_chip_size` but not `locv_core_size`, the software considers the core size to be equal to the `locv_chip_size` value.
- If you specify `locv_core_size` but not `locv_chip_size`, the software ends the LOCV analysis and generates an error message.
- If both `locv_chip_size` and `locv_core_size` are not specified, the software ends the LOCV analysis and generates an error message.

To set this global variable, use the `set_global` command.

locv_inter_clock_use_worst_derate

```
locv_inter_clock_use_worst_derate
     {true | false}
```

Default: `false`

Determines whether to use the worst derating factor for cross clock domain paths.

When set to `true`, the software uses the chip size for the distance value and a stage count of zero, to derive the LOCV derating factor to use on cross clock domain paths.

When set to `false`, the software derives LOCV derating factors as normal, based on calculated distance and stage values for all paths, including cross clock domain paths.

Note: Paths are not considered cross clock domains paths if the:

- Launch and capture clocks are the same.
- Launch and capture clocks have a master clock/generated clock relationship.

To set this global variable, use the `set_global` command.

locv_stage_count_with_IO

```
locv_stage_count_with_IO
     {true | false}
```

Default: `true`

Determines whether I/O cells are considered in the stage count. When set to `true`, the software includes I/O cells in the stage count. When set to `false`, the I/O cells are not included.

To set this global variable, use the `set global` command.

report_precision

`report_precision` *`integer`*

Default: 3

Controls the number of digits appearing after the decimal point in the `report_timing`, `report_net`, `report_cell_instance_timing`, and `report_clocks` reports. The maximum value of *`integer`* is 8.

To set this global variable, use the `set_global` command.

report_timing_format

`report_timing_format {`*`column_list`*`}`

Specifies a `report_timing` format. See report_timing `-format` for more information.

To set this global variable, use the `set global` command.

For example, the following command tells `report_timing` to only list the instance, arc, delay, slew, arrival times in the report:

```
set_global report_timing_format {instance arc delay slew arrival}
```

timing_allow_input_delay_on_clock_source

```
timing_allow_input_delay_on_clock_source
     {true | false}
```

Default: `false`

When set to `true` allows you to apply input delay constraints on a pin where a clock was previously asserted.

To set this global variable, use the `set_global` command.

timing_apply_default_primary_input_assertion

```
timing_apply_default_primary_input_assertion
     {true | false}
```

Default: `true`

When set to `true`, primary input and bidirectional ports that do not have an explicit arrival time specified (through the `set_input_delay` constraint), are provided a default arrival time. The arrival is referenced to an internal default clock that is indicated by the @ sign in various timing reports.

If a `set_input_delay` constraint is assigned to the input port, but the `-clock` parameter was not specified, the software assumes the constraint to be with reference to the default clock.

When set to `false`, inputs ports with no explicit `set_input_delay` are unconstrained for sequential timing analysis.

To set this global variable, use the `set global` command.

timing_apply_exceptions_to_data_check_related_pin

```
timing_apply_exceptions_to_data_check_related_pin
     {true | false}
```

Default: `false`

When set to `true`, any false path assertion which blocks the data path to the related (`-from`) pin of the data-to-data check also causes the check to be disabled.

1) `set_data_check -from portA -to portB`

2) `set_false_path -to portA`

Data check 1 is disabled only when this global is set to `true`.

When set to `false`, only false path exceptions that block the clock reference signal at the related pin affect the data-to-data check (that is, `set_false_path -to`, `-rise_to`, `-fall_to`, with a clock waveform argument).

3) `set_data_check -from portA -to portB`

4) `set_false_path -to [get_clocks CLKA]`

* Path to `portA` is clocked by `CLKA`.

Data check 3 is always disabled; it is not affected by the state of the global.

To set this global variable, use the `set global` command.

timing_build_all_hierarchical_pins

```
timing_build_all_hierarchical_pins
     {true | false}
```

Default: `false`

When set to `true`, includes hierarchical pins in the timing report.

To set this global variable, use the `set global` command.

Note: Setting this global to `true` uses more memory than flat reporting (`false`).

For example:

- The following flat timing report is generated when you specify:

```
set_global timing_build_all_hierarchical_pins false
```

```
Path 1: MET Setup Check with Pin TL2/L1/RX/CK
Endpoint:   TL2/L1/RX/D (v) checked with  leading edge of 'PH1'
Beginpoint: TL1/RX/Q    (v) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.000
- Setup                         0.180
+ Phase Shift                  10.000
= Required Time                 9.820
- Arrival Time                  0.143
= Slack Time                    9.677
     Clock Rise Edge                      0.000
     = Beginpoint Arrival Time            0.000
     +-------------------------------------------------------------------+
     |     Pin     |  Edge |  Net  |  Cell   | Delay | Arrival | Required |
     |             |       |       |         |       |  Time   |   Time   |
     |-------------+-------+-------+---------+-------+---------+----------|
     | clk         |   ^   | clk   |         |       |   0.000 |    9.677 |
     | TL1/RX/CK   |   ^   | clk   | DFFHQX1 | 0.000 |   0.000 |    9.677 |
     | TL1/RX/Q    |   v   | net_1 | DFFHQX1 | 0.143 |   0.143 |    9.820 |
     | TL2/L1/RX/D |   v   | net_1 | DFFHQX1 | 0.000 |   0.143 |    9.820 |
     +-------------------------------------------------------------------+
```

- The following hierarchical timing report is generated when you specify:

```
set_global timing_build_all_hierarchical_pins true

Path 1: MET Setup Check with Pin TL2/L1/RX/CK
Endpoint:   TL2/L1/RX/D (v) checked with  leading edge of 'PH1'
Beginpoint: TL1/RX/Q    (v) triggered by  leading edge of 'PH1'
Other End Arrival Time          0.000
- Setup                         0.180
+ Phase Shift                  10.000
= Required Time                 9.820
- Arrival Time                  0.143
= Slack Time                    9.677
     Clock Rise Edge                      0.000
     = Beginpoint Arrival Time            0.000
    +-------------------------------------------------------------------+
     |     Pin      |  Edge |  Net  |  Cell   | Delay | Arrival | Required |
     |              |       |       |         |       |  Time   |   Time   |
    |--------------+-------+-------+---------+-------+---------+----------|
     | clk          |   ^   | clk   |         |       |   0.000 |    9.677 |
     | TL1/l1_clk   |   ^   | clk   | breg_l1 |       |   0.000 |    9.677 |
     | TL1/RX/CK    |   ^   | clk   | DFFHQX1 | 0.000 |   0.000 |    9.677 |
     | TL1/RX/Q     |   v   | net_1 | DFFHQX1 | 0.143 |   0.143 |    9.820 |
     | TL1/l1_out   |   v   | net_1 | breg_l1 |       |   0.143 |    9.820 |
     | TL2/l2_in    |   v   | net_1 | breg_l2 |       |   0.143 |    9.820 |
     | TL2/L1/l1_in |   v   | net_1 | breg_l1 |       |   0.143 |    9.820 |
     | TL2/L1/RX/D  |   v   | net_1 | DFFHQX1 | 0.000 |   0.143 |    9.820 |
    +-------------------------------------------------------------------+
```

timing_case_analysis_for_icg_propagation

```
timing_case_analysis_for_icg_propagation
     {true | false}
```

Default: `false`

Determines whether constant propagation continues through integrated clock gating (ICG) cells. By default (`false`), constant propagation stops at the ICG cell. When set to `true`, the constant propagates through the ICG cell. If a low logic value at the enable input pin disables the ICG cell, then the low logic value propagates to the fanout of the ICG cell.

Because ICG cells are implemented with level sensitive latches and are sequential by nature, you must set the `timing_case_analysis_for_sequential_propagation` global to `true`, in order to use the `timing_case_analysis_for_icg_propagation` global.

For example:

```
set_global timing_case_analysis_for_sequential_propagation true
set_global timing_case_analysis_for_icg_propagation true
```

To set this global variable, use the `set_global` command.

timing_case_analysis_for_sequential_propagation

```
timing_case_analysis_for_sequential_propagation
     {true | false}
```

Default: `false`

When set to `true`, calculates constants on the outputs of sequential elements. Each time you change the setting of this global, a complete timing update is required and incremental updates are abandoned. If the outputs evaluate to a constant, check arcs and delay arcs are disabled regardless of the setting.

To set this global variable, use the `set global` command.

Note: The `timing_case_analysis_for_sequential_propagation` global should not be used with the `setAnalysisMode` command. Use this global only when the `setAnalysisMode` command is not available. When the `setAnalysisMode` command is available, use the `-sequentialConstProp` parameter to determine whether constants will be propagated through sequential elements.

timing_clock_phase_propagation

```
timing_clock_phase_propagation
     {positive | negative | both}
```

Default: `positive`

Lets you select appropriate clock phases at register clock pins when both positive and negative phases of the same clock signal are seen in the clock network. For example, a register whose clock pin is fed by a two-input XOR gate where one input is a clock port and the other input is a data port. When the value of the data signal of the XOR gate is unknown, the XOR gate can see both positive and negative phases of the clock signal.

`both`	Selects both positive and negative clock phases at the register clock pins.
`positive`	Selects positive clock phases.
`negative`	Selects negative clock phases.

To set this global variable, use the `set global` command.

timing_clock_reconvergence_pessimism

```
timing_clock_reconvergence_pessimism
     {normal | same_transition}
```

Default: `normal`

Note: The `timing_clock_reconvergence_pessimism` global is obsolete. Use the `timing_cppr_transition_sense` global instead.

timing_continue_on_error

```
timing_continue_on_error
     {true | false}
```

Default: `false`

When set to true, directs software to skip the error and continue processing when an error occurs during timing analysis. By default, when an errors occurs during timing analysis, the software stops the process and exits.

To set this global variable, use the `set global` command.

timing_cppr_remove_clock_to_data_crp

```
timing_cppr_remove_clock_to_data_crp
     {true | false}
```

Default: `false`

When set to `true`, removes clock reconvergence pessimism (CPPR) for clock source paths.

To set this global variable, use the `set global` command.

timing_cppr_self_loop_mode

```
timing_cppr_self_loop_mode
     {true | false}
```

Default: `false`

When set to `true` in case of self-loop paths, computes CPPR adjustment by taking the difference between early and late clock arrival time of the common point; the clock pin of the flop. A self-loop path is a path that goes from a clock pin of a flop and ends at the data pin of the same flop. However detecting self-loop paths requires runtime, and is design dependant.

When set to `false`, does not detect self-loop paths. In this case, the software computes CPPR adjustment by using the common point as the output pin driving the clock pin of the flop. This might result in better runtime, but less accurate results. The loss of accuracy is the difference between early and late delay of the interconnect portion driving the clock pin.

To set this global variable, use the `set_global` command.

timing_cppr_skip_clock_reconvergence

```
timing_cppr_skip_clock_reconvergence
     {true | false}
```

Default: `false`

Specifies the branch point to use for computing clock path pessimism removal (CPPR) adjustment when there is reconvergence in the clock tree.

When set to `true`, the software uses the farthest branch point from the register clock pins (that is, the branch point closest to the clock root pin where the clocks diverge). When set to `false`, the software uses the branch point closest to the register clock pins where the clocks reconverge.

To set this global variable, use the `set global` command.

timing_cppr_threshold_ps

`timing_cppr_threshold_ps` *`float`*

Default: `20` ps

Specifies the maximum amount of pessimism that clock path pessimism removal (CPPR) analysis is allowed to leave in the path. The unit is pico seconds. Setting this global to a specified value means that all the paths might be reported without having their pessimism removed. This global can be used to speed up CPPR analysis run time. The larger the value of the global, the faster the run time.

To set this global variable, use the `set_global` command.

timing_cppr_transition_sense

```
timing_cppr_transition_sense
     {normal | same_transition | same_transition_expanded}
```

Default: `normal`

Specifies the transition sense of the launching and capturing clocks at the common node, to calculate clock path pessimism removal (CPPR).

When set to `normal`, the CPPR analysis computes the CPPR value even if the launching and capturing clock transitions at the common node are in opposite directions. If the transition sense at the common node are in opposite directions, the pessimism is the minimum of the difference between late and early arrival times for rise and fall clock transitions.

When set to `same_transition`, the CPPR analysis computes the CPPR value only if the launching clock transition is same as the capturing clock transition at the common node. If the transitions are different, the CPPR value is zero.

When set to `same_transition_expanded`, the CPPR analysis computes the CPPR value only if the launching clock transition is the same as the capturing clock transition at the common mode. If the transitions are different, the software searches until it finds a common point where the transitions are the same, and calculates the CCPR value from that point.

To set this global variable, use the `set global` command.

timing_default_opcond_per_lib

```
timing_default_opcond_per_lib
     {true | false}
```

Default: `true`

Determines the default operating condition behavior when there is no explicit user operating condition specified through the `setOpcond` command.

When set to `true`, each timing library uses its own internally-defined default operating condition specification.

When set to `false`, the default operating condition of the master library (that is, the first library in the library search list) is used to set the PVT operation point for the entire design. All timing libraries are accessed and derated to that operating point.

To set this global variable, use the `set_global` command.

timing_disable_bidi_output_timing_checks

```
timing_disable_bidi_output_timing_checks
     {true | false}
```

Default: `false`

When set to `true`, the software disables timing checks on the output side of bidirectional pins. When set to `false`, timing checks on both the input and output sides of the bidirectional pin are analyzed.

To set this global variable, use the `set global` command.

timing_disable_bus_contention_check

```
timing_disable_bus_contention_check
     {true | false}
```

Default: `false`

When set to `true`, disables propagation of maximum delay along three state disable timing arcs and minimum delay along three state enable arcs. If you know that bus contention conditions do not occur in your design, disable these checks by setting these globals to `true`.

Checks for setup and hold violations in three-state bus designs. Checks the worst delay path to ensure that the latched signals are stable and not unknown values - X. Ensures proper timing checks for bus contention.

To set this global variable, use the `set global` command.

timing_disable_clockgating_checks

```
timing_disable_clockgating_checks
     {true | false}
```

Default: `false` (performs checks)

Important

Global variable "`timing_disable_clockgating_checks`" is obsolete and has been replaced by the following new globals, which allow more precise control over which types of timing checks are being disabled:

- `timing_disable_clock_gating_checks`
- `timing_disable_inferred_clock_gating_checks`
- `timing_disable_non_sequential_checks`
- `timing_disable_user_data_to_data_checks`
- `timing_disable_library_data_to_data_checks`

timing_disable_clock_gating_checks

```
timing_disable_clock_gating_checks
     {true | false}
```

Default: `false`

When set to `true`, disables all clock gating checks, including those declared in the library on integrated clock gating cells, and those that are inferred on combinational gates in the clock path.

To set this global variable, use the `set global` command.

Note: The `timing_disable_clock_gating_checks` global variable should not be used with the `setAnalysisMode` command. Use this global variable only when the `setAnalysisMode` command is not available. When the `setAnalysisMode` command is available, use the `-clockGatingCheck` parameter to control whether clock gating checks are performed.

timing_disable_clockperiod_checks

```
timing_disable_clockperiod_checks
     {true | false}
```

Default: `false` (performs checks)

When set to `true`, disables timing model clock period checks during timing analysis.

To set this global variable, use the `set global` command.

timing_disable_floating_bus_check

```
timing_disable_floating_bus_check
     {true | false}
```

Default: `false`

When set to `true`, disables propagation of minimum delay through three state disable timing arcs and maximum delay through three state enable arcs. These checks are only valid during floating bus conditions.

To set this global variable, use the `set global` command.

timing_disable_inferred_clock_gating_checks

```
timing_disable_inferred_clock_gating_checks
     {true | false}
```

Default: `false` (performs checks)

When set to `true`, disables clock gating checks that are inferred on combinational elements in the clock path. Explicit clock gating checks that are described in the timing library are not affected by this global variable.

To set this global variable, use the `set global` command.

timing_disable_internal_inout_cell_paths

```
timing_disable_internal_inout_cell_paths
     {true | false}
```

Default: `true`

When set to `false`, enables internal bidirectional feedback paths that are completely contained in one instance.

To set this global variable, use the `set_global` command.

timing_disable_internal_inout_net_arcs

```
timing_disable_internal_inout_net_arcs
     {true | false}
```

Default: `false`

When set to `true`, this global disables internal bidirectional feedback paths that span multiple instances.When set to `false`, this global enables internal bidirectional feedback paths, which are feedback paths that you might not want to use during analysis and optimization.

This global applies only to paths that span multiple instances. The global has no impact on internal feedback paths that are completely contained in one instance.

To set this global variable, use the `set_global` command.

timing_disable_lib_pulsewidth_checks

```
timing_disable_lib_pulsewidth_checks
     {true | false}
```

Default: `false` (performs checks)

When set to `true`, disables timing model pulse width checks during timing analysis.

To set this global variable, use the `set global` command.

timing_disable_library_data_to_data_checks

```
timing_disable_library_data_to_data_checks
     {true | false}
```

Default: `false`

When set to `true`, disables data-to-data checks that are coded in the library as non-sequential timing checks. Constraints that are asserted by the `set_data_check` command are not affected by this global variable.

To set this global variable, use the `set global` command.

timing_disable_netlist_constants

```
timing_disable_netlist_constants
     {true | false}
```

Default: `false`

When set to `true`, ignores constants defined in the Verilog netlist. This includes all uses of the in-place constant notation, such as `1'b0`, and the use of power and ground nets. Constants from `set_case_analysis` assertions and from tie-hi and tie-lo cells are still recognized.

When set to `false`, the software recognizes all constants.

To set this global variable, use the `set_global` command.

timing_disable_nochange_checks

```
timing_disable_nochange_checks
     {true | false}
```

Default: `false` (performs checks)

When set to `true`, disables no change timing model checks during timing analysis.

To set this global variable, use the `set global` command.

timing_disable_non_sequential_checks

```
timing_disable_non_sequential_checks
     {true | false}
```

Default: `false`

When set to `true`, disables the timing arcs between any data-to-clock or clock-to-clock checks which have a `timing_type` attribute equal to one of the following:

- `non_seq_setup_rising`
- `non_seq_setup_falling`
- `non_seq_hold_rising`
- `non_seq_hold_falling`

To set this global variable, use the `set global` command.

timing_disable_recovery_removal_checks

```
timing_disable_recovery_removal_checks
     {true | false}
```

Default: `false` (performs checks)

When set to `true`, disables recovery and removal timing model checks during timing analysis.

To set this global variable, use the `set_global` command.

timing_disable_report_header_info

```
timing_disable_report_header_info {true | false}
```

Default: `false`

Controls whether timing reports are generated using a common report header.

When set to `false`, the software generates timing reports with a report header that includes the following information:

- Encounter software version number
- Platform on which the software is being run
- Time when the report was generated
- Command used to generate the report

Any report-specific banner information is included in the same header box, but listed after the common header information.

When set to `true`, the software generates backward-compatible timing reports without the additional header information.

To set this global variable, use the `set_global` command.

timing_disable_set_case_analysis

```
timing_disable_set_case_analysis
     {true | false}
```

Default: `false`

When set to `true`, blocks the application of constants from the timing constraints file. Constants that are introduced through the netlist and the timing library are not affected by this global.

To set this global variable, use the `set global` command.

Note: The `timing_disable_set_case_analysis` global variable should not be used with the `setAnalysisMode` command. Use this global variable only when the `setAnalysisMode` command is not available. When the `setAnalysisMode` command is available, use the `-caseAnalysis` parameter to control whether constants are applied.

timing_disable_skew_checks

```
timing_disable_skew_checks
     {true | false}
```

Default: `false`

When set to `true`, disables library skew checks for timing analysis.

To set this global variable, use the `set global` command.

timing_disable_test_signal_arc

```
timing_disable_test_signal_arc
     {true | false}
```

Default: `false`

When set to `true`, timing analysis will not analyze the signal arcs coming from or going to the test pin. Such pins are marked in the timing library with either the `test_scan_in`, the `test_scan_enable` attribute for the input pin, the `test_scan_clock` attribute for the input clock pin, or the `test_output_only` attribute for the output pin.

The following are the equivalent Liberty and TLF pin attributes:

Table 35-1 Liberty and TLF Pin Attributes

Type	Liberty	TLF
enable	`test_scan_enable`	`SCAN_PINTYPE` (ENABLE)
input	`test_scan_in`	`SCAN_PINTYPE` (INPUT)
input clock	`test_scan_clock`	`SCAN_PINTYPE` (CLOCK)
output	`test_output_only`	`SCAN_PINTYPE` (OUTPUT)

To set this global variable, use the `set global` command.

timing_disable_timing_model_latch_inferencing

```
timing_disable_timing_model_latch_inferencing
     {true | false}
```

Default: `false`

Controls whether latch behavior is inferred for cell descriptions tagged with the Liberty `timing_model_type` attribute, including all values: `abstracted`, `extracted`, or `qtm`.

When set to `false`, the software infers latch behavior from the timing check and timing delay arc relationships in the model's description.

When set to `true`, latch inferencing does not happen for cells with the `timing_model_type` attribute. Explicit Liberty latch modelling constructs must be coded into the description in order to model latch behavior.

This global does not affect Liberty timing model cells that do not contain the `timing_model_type` attribute. Standard cell core libraries generally do not have this attribute specified; therefore, they typically are not be affected by this global setting.

To set this global variable, use the `set global` command.

timing_disable_tristate_disable_arcs

```
timing_disable_tristate_disable_arcs
     {true | false}
```

Default: `false`

When set to `true`, disables all 0/1->Z transitions of tristate arcs during timing analysis. By default, all tristate arcs are enabled.

To set this global variable, use the `set_global` command.

timing_disable_user_data_to_data_checks

```
timing_disable_user_data_to_data_checks
     {true | false}
```

Default: `false`

When set to `true`, disables data-to-data checks that are created by the `set_data_check` command. This global variable does not affect non-sequential timing checks described in the timing library.

To set this global variable, use the `set global` command.

timing_dynamic_loop_breaking

```
timing_dynamic_loop_breaking
     {true | false}
```

Default: `false`

Enables dynamic loop breaking.

By default (`false`), when the timing engine encounters combinational loops in the design, it breaks the loop by disabling a timing arc along the path. The heuristics used in breaking the loop cannot guarantee that a real path through the disabled arc will not be blocked as a consequence. Therefore, users must carefully investigate the arcs that have been disabled to ascertain that the loop break was done safely. If loop breaks are not then added to the master constraint file, they could move to other locations as the design matures.

When set to `true`, the timing system performs a more sophisticated analysis of the design, and ensures that no valid paths are broken inadvertently due to loops.

Note: Setting this global variable to `true` will add additional run time to the analysis.

To set this global variable, use the `set global` command.

timing_enable_clock_path_pessimism_removal

```
timing_enable_clock_path_pessimism_removal
     {true | false}
```

Default: false

Important

Global variable “`timing_enable_clock_path_pessimism_removal`” is obsolete and has been replaced by “`timing_remove_clock_reconvergence_pessimism`”.

timing_enable_data_through_clock_gating

```
timing_enable_data_through_clock_gating
     {true | false}
```

Default: `true`

When set to `false`, blocks signals arriving on the enable of the clock-gating check.

When set to `true`, allows both data and clock signals at the enable pin of a clock-gating check to propagate forward past the check.

To set this global variable, use the `set global` command.

timing_enable_default_delay_arc

```
timing_enable_default_delay_arc
     {true | false}
```

Default: `true`

Controls the behavior of the default arc if multiple delay arcs, each with a different sdf condition, exist between a pair of pins (parallel delay arcs). The default arc is the delay arc with no sdf condition.

If the sdf condition on any arc explicitly evaluates to `true`, that arc is enabled and the default arc is disabled. When there are sdf conditions to cover all possible values of an input variable and none of these sdf conditions explicitly evaluate to `true`, then the default arc is active if this variable is set to `true`. Otherwise, the default arc is inactive.

For example, consider two delay arcs `A->Z` with sdf conditions `B` and `B'`. If a third arc exists between `A->Z` with no condition (default arc), and `B=X`, which means `B` is either a 1 or 0, then only the arcs with the corresponding condition should be considered. The is the behavior when the `timing_enable_default_delay_arc` variable is set to `false`.

If the variable is set to `true`, then the default arc delay, which is the delay of the arc with no condition, is also considered for delay calculation.

To set this global variable, use the `set global` command.

timing_enable_genclk_edge_based_source_latency

```
timing_enable_genclk_edge_based_source_latency
     {true | false}
```

Default: `true`

Controls how the software chooses generated clock source latency paths. Certain specifications of the `create_generated_clock` constraint imply a cause-and-effect relationship between the edges of the master clock and those of the generated clock.

When set to `true`, the software checks whether a path being analyzed has the correct cause-effect relationship consistent with the specified constraint. Paths that are found to be inconsistent are ignored. Only the paths that are in agreement with the required edge relationship are considered for source latency. If no such path exists, the software uses a source latency of `0` ns. This selection is done independently for each edge of the generated clock; therefore the result might be different for each edge.

In addition, edge-to-edge sufficiency checks are performed assuming a Rise-to-Rise, Rise-to-Fall relationship between the master clock and the generated clock with an even number `-divide_by` factor. For the odd `-divide_by` factors, the sufficiency check is performed assuming a Rise-to-Rise, Fall-to-Fall relationship. The sufficiency check is not performed for `-multiply_by` options. This behavior provides a better correlation with the SDC specification semantic requirements.

When set to `false`, the software does not check paths for the correct cause-effect relationship.

To set this global variable, use the `set_global` command.

timing_enable_mmmc_loop_handling

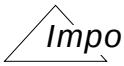
```
timing_enable_mmmc_loop_handling
     {true | false}
```

Default: `true`

When set to `true`, loop breaking is handled independently per analysis view, according to the following rules in order of precedence:

- If an analysis view does not contain a combinational loop, it does not inherit any loop breaking done for other views.
- If a loop exists, and the user has supplied specific case analysis and `set_disable_timing` constraints to break the loop at a certain place, the specified loop break is honored.
- If a loop is still detected, the software automatically selects a point to break the loop, and applies the loop break consistently to any views where the loop exists.

When set to `false`, if a combinational loop is detected in any analysis view, the timing system breaks the loop by disabling an arc along the looping path. These combinational loop breaks are reflected in all analysis views, even if those views by themselves do not contain these looping paths.



Important

Cadence strongly recommends that all combinational loops detected by the software be evaluated, to ensure that the loop is broken at the most ideal location. Automatic loop breaking decisions are made based on how the design is traversed. Small changes in the design can influence where loop breaking decisions are made.

To reduce risk and the need to re-evaluate loop-breaking decisions, it is highly recommended that once combinational loop breaks have been evaluated, the user add constraints to ensure that the loop is always broken in that location.

To set this global variable, use the `set global` command.

timing_enable_multifrequency_latch_analysis

```
timing_enable_multifrequency_latch_analysis
     {true | false}
```

Default: `false`

Enables multi-frequency latch timing analysis of the latch time borrowing for when a data signal coming to a latch is controlled by a clock with a frequency different to a clock of the latch enabling signal.

When set to `true`, the edges of all sending and receiving clocks are analyzed at every latch, which results in less pessimistic, and more detailed timing analysis.

When set to `false`, latch time borrowing analysis uses the worst case relationship between sending and receiving clocks at the latch, which can produce paths with pessimistic slack.

This global variable also enables the use of the input pin of a transparent latch arc (the D pin) as a `-from` argument in the `set_false_path` and `set_multicycle_path` exceptions.

Note: Enabling advanced latch analysis can result in some performance penalty.

To set this global variable, use the `set_global` command.

timing_enable_power_ground_constants

```
timing_enable_power_ground_constants
     {true | false}
```

Default: `true`

Controls whether case analysis is inferred from power and ground rail connections.

When set to `true`, the software recognizes constants that come from power and ground rail connections. When set to `false`, the software ignores constants from these connections.

To set this global variable, use the `set global` command.

timing_enable_preset_clear_arcs

```
timing_enable_preset_clear_arcs
     {true | false}
```

Default: `false`

Determines whether timing arcs are created to model the transition to active state (assertion) of the preset or clear pin, and the subsequent transition of the output to controlled state.

When set to `true`, the software builds these arcs, and the timer automatically analyzes paths through them.

By default (`false`), the software does not build these arcs, because they are usually associated with analyzing the reset mode timing paths. Filtering them prevents unwanted interaction with functional mode timing analysis.

Use the `lib_build_asynch_de_assert_arc` global to control whether input to output arcs from the preset or clear pins transitioning to inactive state are included when the timing system is initialized.

Note: If both the `timing_enable_preset_clear_arcs` and the `lib_build_asynch_de_assert_arc` global variables are set to `true`, all asynchronous arcs are recognized. However, if `timing_enable_preset_clear_arcs` is set to `true` and `lib_build_asynch_de_assert_arc` is set to `false`, only assert asynchronous arcs are recognized.

Because the `timing_enable_preset_clear_arcs` and `lib_build_asynch_de_assert_arc` global variables affect how the timing graph is built, it is important to set the desired state before performing any timing analysis.

To set this global variable, use the `set_global` command.

timing_enable_si_cppr

```
timing_enable_si_cppr
     {true | false}
```

Default: `false`

Enables more accurate CPPR analysis when incremental delays are present and the `timing_remove_clock_reconvergence_pessimism` global variable is set to `true`.

When set to `true`, the incremental delay is not included in the CPPR calculation during setup analysis, but is included in the CPPR calculation during hold analysis.

When set to `false`, the incremental delay is included in both setup and hold CPPR calculations. This can lead to optimistic results for setup analysis.

To set this global variable, use the `set global` command.

timing_enable_simultaneous_setup_hold_mode

```
timing_enable_simultaneous_setup_hold_mode
     {true | false}
```

Default: `false`

Controls whether setup and hold checks are analyzed separately, or together on the same timing graph.

When set to `false` (the default value), the `setAnalysisMode -checkType setup` and `hold` parameters are used to switch the system from analyzing setup slacks to analyzing hold slacks.

When set to `true`, setup and hold slacks are analyzed concurrently. As a consequence, both `-early` and `-late` queries for reporting commands such as `report_timing` can be satisfied immediately, without additional calculation. Simultaneous mode is useful for immediately understanding the consequences to both setup and hold slacks from what-if operations and manual ECOs to the design.

Note: Simultaneous setup and hold mode can be used only for timing analysis; it cannot be used for any physical implementation steps.

Simultaneous setup/hold mode has the following behavior differences, with respect to the default mode:

- `setAnalysisMode -checkType setup` and `hold` do not function in simultaneous mode.
- The `write_timing_windows` command does not generate timing window information using the min and max libraries. Instead, by default, it uses only the max libraries.

To set this global variable, use the `set_global` command.

timing_enable_tristate_clock_gating

```
timing_enable_tristate_clock_gating
     {true | false}
```

Default: `false`

The timing system infers gated-clock checks in the design where a data and clock path intersect at a combinational device. When this global is set to `true`, inferred gated-clock checks are added when clock and data signals converge through the tristate enable and data input of tristate buffers. By default (a setting of `false`), gated-clock checks are not added for these situations.

To set this global variable, use the `set_global` command.

timing_enable_uncertainty_for_pulsewidth_checks

```
timing_enable_uncertainty_for_pulsewidth_checks
     {true | false}
```

Default: `false`

When set to `true`, considers clock uncertainty when performing minimum pulse width checks. Clock uncertainty is considered during checks of both user-defined minimum pulse widths (using the `set_min_pulse_width`), and minimum pulse widths that are defined in the timing model (`.lib`).

Note: Specifying a value of `true` affects the following commands that take minimum pulse width checks into account:

- `report_timing -check_type pulse_width`
- `report_timing -check_clocks`
- `report_min_pulse_width`
- `report_analysis_coverage`

To set this global variable, use the `set_global` command.

timing_extract_model_slew_propagation_mode

```
timing_extract_model_slew_propagation_mode
     {worst_slew | path_based_slew}
```

Default: `path_based_slew`

Specifies the type of slew propagation to use for generating extracted timing model (ETM).

In worst slew propagation mode, the software propagates the worst slew of all the incoming arcs at a converging point for extracting the arcs. This is the recommended mode.

In path based slew propagation mode, the software propagates the actual slew for the path elements for extracting the arcs. This is the default mode.

To set this global variable, use the `set_global` command.

timing_hier_object_name_compatibility

```
timing_hier_object_name_compatibility
     {true | false}
```

Default: `true`

Controls how hierarchical delimiters are interpreted in the search pattern, when the `-hier` parameter is used with the `get_*` collection commands.

When set to `true`, the software interprets the hierarchical delimiter as part of the search pattern itself, and tries to match names that include the delimiter. For example, if you specify:

```
get_cells -hier L1/R*
```

The software searches at every level of the hierarchy for instances with the name `L1/R*`, completely ignoring the hierarchy separator:

When set to `false`, hierarchical delimiters are interpreted as hierarchy separators. For example, if you specify:

```
get_cells -hier L1/R*
```

The software searched for instances with names matching `R*` in the hierarchy level below `L1`:

```
TL2/L1/RX1 TL2/L1/RX2
```

To set this global variable, use the `set_global` command.

timing_ignore_lumped_rc_assertions

```
timing_ignore_lumped_rc_assertions
     {true | false}
```

Default: `true`

When set to `true`, instructs the software to ignore `set_load` and `set_resistance` assertions on internal nets in the design. Parasitic data for internal nets normally should be provided by Encounter RC extraction, or external SPEF annotation. If allowed (a setting of `false`), `set_load` and `set_resistance` values override the actual extracted representation.

Using `set_load` to specify I/O boundary loading on ports is supported by either setting of this global.

To set this global variable, use the `set global` command.

timing_io_use_clock_network_latency

```
timing_io_use_clock_network_latency
     [always | ideal]
```

Default: `ideal`

Controls whether network latency of a reference clock is added or not to the data arrival time on the port. When set to `ideal`, the `set_input_delay` and the `set_output_delay` constraints will not add the network latency of the reference clock to the data arrival time on the port if the clock is in `propagated` mode.

When this global is set to a`lways`, network latency is added to the data arrival time on the port regardless of the clock propagation mode.

To set this global variable, use the `set global` command.

timing_library_genclk_use_group_name

```
timing_library_genclk_use_group_name
     {true | false}
```

Default: `false`

When set to `true`, the software uses the generated_clock group name when creating a generated clock from a library-generated clock group. When set to `false`, the software uses the pin name when creating the generated clock.

For example, assume the cell for instance `a/b` in the timing library contained the following generated clock group:

```
generated_clock(genclk2) {
    clock_pin : clk1;
    master_pin : “int/clk”;
    multiplied_by : 2;
    duty_cycle : 50.0;
}
```

By default (`false`), the software creates a generated clock with the following name:

```
a/b/int/clk
```

If the `timing_prefix_module_name_with_library_genclk` global variable is set to `true`, and you set this global variable to `true`, the software creates a generated clock with the following name:

```
a/b/genclk2
```

If the `timing_prefix_module_name_with_library_genclk` global variable is set to `false`, and you set this global variable to `true`, the software creates a generated clock with the following name:

```
genclk2
```

To set this global variable, use the `set global` command.

timing_library_zero_negative_timing_check_arcs

```
timing_library_zero_negative_timing_check_arcs
     {true | false}
```

Default: `false`

When set to `true`, converts any negative timing check values in the timing library to zero.

timing_path_groups_for_clocks

```
timing_path_groups_for_clocks
     {true | false}
```

Default: `false`

When set to `true`, enables automatic path group creation for the create_clock and the create_generated_clock commands. Creates a new path group to bring together the endpoints related to the clock. Set this global before reading constraints.

Use the `timing_path_groups_for_clocks` global variable before importing the design data.

To set this global variable, use the `set_global` command.

timing_prefix_module_name_with_library_genclk

```
timing_prefix_module_name_with_library_genclk
     {true | false}
```

Default: `true`

When set to `true`, the software appends the instance name to the clock pin name when creating a generated clock. When set to `false`, the software uses only the clock pin name when creating a generated clock.

For example, assume the cell for instance `a/b` in the timing library contained the following generated clock group:

```
generated_clock(genclk2) {
    clock_pin : clk1;
    master_pin : “int/clk”;
    multiplied_by : 2;
    duty_cycle : 50.0;
}
```

By default (`true`), the software creates a generated clock with the following name:

```
a/b/int/clk
```

If you set the global to `false`, the software creates a generated clock with the following name:

```
int/clk
```

If you set this global to `false`, and the `timing_library_genclk_use_group_name` global variable is set to `true`, the software creates a generated clock with the following name:

```
genclk2
```

To set this global variable, use the `set_global` command.

timing_propagate_latch_data_uncertainty

```
timing_propagate_latch_data_uncertainty
     {true | false}
```

Default: `false`

When set to `true`, uses the clock phase associated with a flush latch's data pin as the "from" clock phase for downstream uncertainty timing calculations

When set to `false`, the downstream uncertainty calculations are based on the clock phase of the flush latch's clock signal.

To set this global variable, use the `set global` command.

timing_recompute_sdf_in_setuphold_mode

```
timing_recompute_sdf_in_setuphold_mode
     {true | false}
```

Controls the recomputing of SDF delays when the software is in simultaneous setup and hold analysis mode.

When the software is not in simultaneous setup and hold analysis mode, only one corner of data is stored at a time; therefore additional delay calculation is required for `write_sdf`. When the software is in simultaneous setup and hold analysis mode, both sets of data are in memory at the same time, and additional delay calculation is not necessary.

If set to `false`, `write_sdf` will write out the delays from memory, instead of calling delay calculation.

Note: This global variable must be used with `timing_enable_simultaneous_setup_hold_mode` `true`.

To set this global variable, use the `set global` command.

timing_reduce_multi_drive_net_arcs

```
timing_reduce_multi_drive_net_arcs
     {true | false}
```

Default: `true`

Controls the reduction of the number of net arcs created for timing analysis for nets driven by parallel buffers. If a net is driven by n parallel buffers and is connected to `m` sink pins, the total number of net arcs created is `n*m`. For example, if `n` is `100` and `m` is `20000,` as is the case in some clock network schemes, the number of net arcs created for timing analysis is two million. This results in a runtime and memory penalty.

This global lets you reduce the number of net arcs created. When the global is set, net arcs are only created from one driver or from the drivers for which you have set the constraints. In the example, the number of net arcs will be `20000`.

If you specify constraints on the net driven by parallel buffers, it might result in runtime and memory degradation. By specifying timing constraints through nets, the constraints are moved to its drivers. This leads to the creation of net arcs from all the drivers, disabling net arc reduction. In such a case, select one driver of the net and use that driver to specify the constraints.

To set this global variable, use the `set global` command.

timing_reduce_multi_drive_net_arcs_threshold

`timing_reduce_multi_drive_net_arcs_threshold` *`int`*

Default: `10000`

Sets a threshold number used by the tool to trigger the reduction of timing arcs of nets driven by parallel buffers. The reduction takes place if the number of arcs is greater than the threshold value and if the timing_reduce_multi_drive_net_arcs global is set to `true`.

To set this global variable, use the `set global` command.

timing_remove_clock_reconvergence_pessimism

```
timing_remove_clock_reconvergence_pessimism {true | false}
```

Default: `false`

Enables Clock Path Pessimism Removal (CPPR) analysis during timing analysis. When set to `true`, the timing analysis commands remove pessimism from slack calculation. Clock path pessimism is the difference of launching and capturing clock arrival times at the common node. This pessimism can be due to difference in minimum and maximum cell delays along the clock paths or due to the reconvergent clock paths. If the launching and capturing registers are triggered by the same clock and there is no common node between the launching and capturing clock paths, the pessimism is the difference between late and early clock source latencies.

To set this global variable, use the `set_global` command.

Note: The `timing_remove_clock_reconvergence_pessimism` global should not be used with the `setAnalysisMode` command. Use this global only when the `setAnalysisMode` command is not available. When the `setAnalysisMode` command is available, use the `-cppr` parameter to set or reset the CPPR analysis.

timing_report_launch_clock_path

```
timing_report_launch_clock_path {true | false}
```

Default: `false`

Controls whether `report timing` reports the detailed clock latency path.

When set to `false`, the `report_timing` command reports the cumulative ideal or propagated latency in a single line in each report section. The reported delay is a summary of the delays on the path.

When set to `true`, the `report_timing` command reports the launching clock path.

Note: If you specify `report_timing -path_type full_clock`, both the detailed launching and capturing clock paths are reported, regardless of the setting of this global variable.

To set this global variable, use the `set_global` command.

timing_report_paths_through_sequential_arcs

```
timing_report_paths_through_sequential_arcs
     {true | false}
```

Default: `false`

Controls whether the `report_timing` `-from` and `-through` parameters are allowed to reference pins or ports that strictly belong to the clock path.

When set to `true`, pins or ports strictly belonging to clock paths can be referenced with the `-from` and `-through` parameters. When set to `false`, only data path pins or port arguments can be specified with the `-from` and `-through` parameters.

For example, the following specification is accepted by the software when this global variable is set to `true`, but is not legal when the global variable is set to `false`:

```
report_timing -from myClockPort -through myClockTreeGate/Y  -to endReg/D
```

The software always allows clock waveforms as `-from` arguments, and considers the clock pin of the launching register to be the start of the data path, so it is considered a legal `-through` argument. For example,

```
report_timing -from myClockWaveform -through startReg/CK -to endReg/D
```

If the clock path also functions as a source clock data path (that is, there is combinational path from the clock definition to a data check), you can use `-from` and `-through` to reference points along this specific path. For example,

```
report_timing -from myClockPort -through myClockTreeGate/Y -to clkSrcPathEnd/D
```

To set this global variable, use the `set_global` command.

timing_report_unconstrained_paths

```
timing_report_unconstrained_paths
     {true | false}
```

Default: `false`

When set to true, the `report_timing` command reports unconstrained paths if it cannot find a constrained path to report. When set to false, the report_timing command only reports constrained paths.

Important

The behavior of `timing_report_unconstrained_paths` is not the same as the behavior of `report_timing -unconstrained`. The `report_timing -unconstrained` parameter always reports unconstrained paths, even if a constrained path can be reported. In the same situation, the `timing_report_unconstrained_paths` global reports the constrained path.

To set this global variable, use the `set global` command.

timing_self_loop_paths_no_skew

```
timing_self_loop_paths_no_skew
     {true | false}
```

Default: `false`

When set to `true`, factors the skew on self loops. The software ignores the skew on the clock pin for hold analysis for data paths originating on sequential elements (for example, Q-pin of a flip flop) and ending on the data pins of the same element (D-pin of the same flip flop). Use this variable to eliminate pessimism due to skew when the launch and the capture edge of the clock are the same.

To set this global variable, use the `set_global` command.

Note: The `timing_self_loop_paths_no_skew` global should not be used with the `setAnalysisMode` command. Use this global only when the `setAnalysisMode` command is not available. When the `setAnalysisMode` command is available, use the `-timingSelfLoopsNoSkew` parameter to enable skew factoring on self loops.

timing_self_loop_paths_no_skew_max_depth

`timing_self_loop_paths_no_skew_max_depth` *`value`*

Default: 10

Identifies self-loop paths at the specified depth.

To set this global variable, use the `set_global` command.

For example:

```
set_global timing_self_loop_paths_no_skew_max_depth 20
```

Lets the timer traverse 20 timing arcs (10 stages of logic) before abandoning the search for a self loop.

timing_self_loop_paths_no_skew_max_slack

`timing_self_loop_paths_no_skew_max_slack` *`value`*

Default: `0`

Identifies self-loop paths at the specified threshold.

To set this global variable, use the `set_global` command.

For example:

```
set_global timing_self_loop_paths_no_skew_max_slack 1.0
```

In this case, all timing endpoints with a worst slack less than `1.0` are considered for self loop skew removal.

timing_set_scaling_for_negative_checks

```
timing_set_scaling_for_negative_checks
     {default | divider | multiplier}
```

Default: Negative check * (2.0 - derating factor), where the derating factor is the value that you set using the `set_timing_derate` command.

Modifies the scaling of the negative timing analysis check value. This variable does not affect the positive checks. When set to `divider`, divides the negative check value by the derating factor. When set to `multiplier`, multiplies the negative check value by the derating factor.

To set this global variable, use the `set global` command.

timing_set_scaling_for_negative_delays

```
timing_set_scaling_for_negative_delays
     {default | divider | multiplier}
```

Default: Negative delay * (2.0 - derating factor), where the derating factor is the value that you set using the `set_timing_derate` command.

Modifies the scaling of the negative delay value to be used during timing analysis. This variable does not affect the positive delay. When set to `divider`, divides the negative delay value by the derating factor. When set to `multiplier`, multiplies the negative delay value by the derating factor.

To set this global variable, use the `set_global` command.

Currently, constraints are used during timing in the flattened mode, so the internal ILM instances are seen instead of the LEF pins of the ILMs. Therefore, reading the bounding box constraints causes errors without this variable set. Notice that if you want to see the LEF pins of the ILM in GUI, the design must be in the unflattened mode.

timing_suppress_ilm_constraint_mismatches

```
timing_suppress_ilm_constraint_mismatches
     {true | false}
```

Default: `false`

When set to `true`, the software suppresses all error and warning messages related to objects not found when loading SDC constraint files for the ILM flow.

In the current ILM flow, SDC constraints originally specified against the complete flat netlist for the design do not get filtered when being applied against the ILM view of the design. Constraints that reference parts of the design that were pruned cause warnings and errors during constraint loading.

To set this global variable, use the `set global` command.

timing_use_latch_time_borrow

```
timing_use_latch_time_borrow
     {true | false}
```

Default: `true`

When set to `false`, does not consider time borrowing during timing analysis. Time borrowing is the amount of time borrowed by a previous logic.

To set this global variable, use the `set_global` command.

Note: The `timing_use_latch_time_borrow` global should not be used with the `setAnalysisMode` command. Use this global only when the `setAnalysisMode` command is not available. When the `setAnalysisMode` command is available, use the `-timeBorrowing` parameter to control time borrowing consideration.

write_global_slack_worst_trigger_path_on_clocks

```
write_global_slack_worst_trigger_path_on_clocks
     {true | false}
```

Default: `false`

Specifies whether the `write global slack report` command reports worst slack on the register clock pins or not. When set to `true`, the `write_global_slack_report` command reports worst slack due to triggering paths.

A CLK pin might drive a Q pin of the same register, and have a setup or hold check arc with respect to the D pin. In the default setting, when the global is set to false, the `write_global_slack_report` command reports no slacks along the clock paths. When the global is set to `true`, the command will report the worst slack on register clock pins due to the related trigger paths and not due to the related timing checks.

To set this global variable, use the `set global` command.

36

Timing Optimization Commands

checkFootPrint

```
checkFootPrint
```

Checks the timing library for missing footprint functions or timing information.

Parameters

None

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Cell Footprints"

createBasicPathGroups

```
createBasicPathGroups
```

Creates the five most commonly used path groups:

- `reg2reg`
- `in2reg`
- `reg2out`
- `in2out`
- `clkgate`

Path groups are supported natively by the Common Timing Engine (CTE). Use the `reset_path_group` command to delete a path group. Use the `setPathGroupOptions` command to set or change a path group option.

You can also create path groups by using the `group_path` command in an SDC file.

Important

To use path groups, you must enable them by specifying the following command:

```
setAnalysisMode -honorClockDomains false
```

For more information, see `setAnalysisMode` in the “Timing Analysis (Common Timing Engine) Commands” chapter.

Parameters

None

deleteBufferTree

```
deleteBufferTree
     [-selNetFile fileName]
     [-excNetFile fileName]
     [-footprint footprintType]
     [-noCreateAssign | -allowCreateAssign]
```

Removes all buffers of footprint type specified.

Note: This command does not delete feedthrough buffers inserted by `SOC-E: *FE_FEEDX_*`.

Parameters

`-excNetFile` *`fileName`*

Specifies the input file that contains the names of nets to exclude from buffer removal. If a net appears in the files specified by both `-excNetFile` and `-selNetFile`, it is excluded.

`-footprint` *`footprintName`*

Specifies the footprint of buffers to remove. If you do not specify this parameter, the Encounter® software removes buffer instances with one input and one output terminal, and whose timing arc is not `SPECIAL`.

`-noCreateAssign|-allowCreateAssign`

Specifies whether `Assign` nets are allowed between two ports on a module, creating feedthroughs. If no feedthroughs are allowed, at least one buffer is retained on the net connecting the ports.
Default: `-noCreateAssign`

`-selNetFile` *`fileName`*

Specifies the file that contains the names of hierarchical nets to include during buffer removal. The Encounter software excludes all other nets. If a net appears in the files specified by both `-excNetFile` and `-selNetFile`, it is excluded.

Example

The following command removes all buffers of footprint type B_1P:

```
deleteBufferTree -footprint B_1P
```

getLatencyFile

`getLatencyFile`

Returns the name of the latency file used by the timing engine. If no latency file name is set, `getLatencyFile` returns nothing, but if you have already run `skewClock`, this command reports the default file name. If you have specified or changed the latency file name by using the `setLatencyFile` command, `getLatencyFile` reports this name.

Parameters

None

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Performing Optimization Before Clock Tree Synthesis"

getOptMode

```
getOptMode
     [-addInst]
     [-addInstancePrefix]
     [-addNetPrefix]
     [-addPortAsNeeded]
     [-bufferAssignNets]
     [-congOpt]
     [-considerNonActivePathGroup]
     [-criticalRange]
     [-critPathCellYield]
     [-deleteInst]
     [-downsizeInst]
     [-drcMargin]
     [-effort]
     [-fixDrc]
     [-fixFanoutLoad]
     [-fixHoldAllowSetupTnsDegrade]
     [-holdFixingEffort]
     [-holdSdfFile]
     [-holdTargetSlack]
     [-keepPort]
     [-leakagePowerEffort]
     [-maxDensity]
     [-moveInst]
     [-optimizeConstantNet]
     [-optimizeFF]
     [-optimizeNetsAcrossDiffVoltPDs]
     [-postRouteAllowOverlap {true | false}]
     [-postRouteSiAware {true | false}]
     [-preserveAssertions]
     [-preserveModuleFunction]
     [-reclaimArea]
     [-resizeShifterAndIsoInsts]
     [-restruct]
     [-setupSdfFile]
     [-setupTargetSlack]
     [-simplifyNetlist]
     [-sizeOnlyFile]
     [-swapPin]
     [-useConcatDefaultsPrefix]
     [-usefulSkew]
     [-verbose]
     [-yieldEffort]
     [-quiet]
```

Returns the following information about `setOptMode` parameters in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the Encounter software returns values for all of the `setOptMode` parameters.

Parameters

`parameter_names`	Returns information for the specified parameter. You can specify one or more parameters. See `setOptMode` for descriptions of the parameters you can specify.
`-quiet`	Returns the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software returns the current settings of all `setOptMode` parameters in Tcl list format.

Examples

- The following command returns the current setting of the `-addInst` parameter:

  ```
  getOptMode -addInst
  ```

  The software returns the following information:

  ```
  -addInst true                # bool, default=true
  true
  ```

- The following command returns the current settings for all `setOptMode` parameters:

  ```
  getOptMode
  ```

  The software returns the following information:

  ```
  -addInst true                              # bool, default=true
  -addInstancePrefix {}                      # string, default=""
  -addNetPrefix {}                           # string, default=""
  -addPortAsNeeded true                      # bool, default=true
  -bufferAssignNets false                    # bool, default=false
  -congOpt false                             # bool, default=false
  -considerNonActivePathGroup false          # bool, default=false
  ```

```
-criticalRange 0.2                          # float, default=0.2, min=0.000000,
max=1.000000
-deleteInst true                              # bool, default=true
-downsizeInst true                            # bool, default=true
-drcMargin 0                               # float, default=0, min=-1000.000000,
max=0.700000
-effort high                                  # enums={low high}, default=high
-fixDrc true                                  # bool, default=true
-fixFanoutLoad false                          # bool, default=false
-fixHoldAllowSetupTnsDegrade true             # bool, default=true
-holdSdfFile {}                               # string, default=""
-holdTargetSlack 0                            # float, default=0
-keepPort {}                                  # string, default=""
-leakagePowerEffort none                      # enums={none low high min},
default=none
-maxDensity 0.95                              # float, default=0.95
-moveInst true                                # bool, default=true
-optimizeConstantNet true                     # bool, default=true
-optimizeFF true                              # bool, default=true
-optimizeNetsAcrossDiffVoltPDs true           # bool, default=true
-postRouteAllowOverlap true                   # bool, default=true
-postRouteSiAware false                       # bool, default=false
-preserveModuleFunction false                 # bool, default=false
-resizeShifterAndIsoInsts false               # bool, default=false
-restruct true                                # bool, default=true
-setupSdfFile {}                              # string, default=""
-setupTargetSlack 0                           # float, default=0
-simplifyNetlist false                        # bool, default=false
-useConcatDefaultsPrefix true                 # bool, default=true
-usefulSkew false                             # bool, default=false
-verbose false                                # bool, default=false
-yieldEffort none                           # enums={none low high}, default=none
```

```
{addInst true} {addInstancePrefix {}} {addNetPrefix {}} {addPortAsNeeded true}
{bufferAssignNets false} {congOpt false} {considerNonActivePathGroup false}
{criticalRange 0.2} {deleteInst true} {downsizeInst true} {drcMargin 0} {effort
high} {fixDrc true} {fixFanoutLoad false} {fixHoldAllowSetupTnsDegrade true}
{holdSdfFile {}} {holdTargetSlack 0} {keepPort {}} {leakagePowerEffort none}
{maxDensity 0.95} {moveInst true} {optimizeConstantNet true} {optimizeFF true}
{optimizeNetsAcrossDiffVoltPDs true} {postRouteAllowOverlap true}
{postRouteSiAware false} {preserveModuleFunction false}
{resizeShifterAndIsoInsts false} {restruct true} {setupSdfFile {}}
{setupTargetSlack 0} {simplifyNetlist false} {useConcatDefaultsPrefix true}
{usefulSkew false} {verbose false} {yieldEffort none}
```

- The following command returns the current setting of the `-addInst` parameter in Tcl list format only:

  ```
  getOptMode -addInst -quiet
  ```

  The software returns the following information:

  ```
  true
  ```

- The following command returns the current settings for all `setOptMode` parameters in Tcl list format only:

  ```
  getOptMode -quiet
  ```

  The software returns the following information:

  ```
  {addInst true} {addInstancePrefix {}} {addNetPrefix {}} {addPortAsNeeded true}
  {bufferAssignNets false} {congOpt false} {considerNonActivePathGroup false}
  {criticalRange 0.2} {deleteInst true} {downsizeInst true} {drcMargin 0} {effort
  high} {fixDrc true} {fixFanoutLoad false} {fixHoldAllowSetupTnsDegrade true}
  {holdSdfFile {}} {holdTargetSlack 0} {keepPort {}} {leakagePowerEffort none}
  {maxDensity 0.95} {moveInst true} {optimizeConstantNet true} {optimizeFF true}
  {optimizeNetsAcrossDiffVoltPDs true} {postRouteAllowOverlap true}
  {postRouteSiAware false} {preserveModuleFunction false}
  {resizeShifterAndIsoInsts false} {restruct true} {setupSdfFile {}}
  {setupTargetSlack 0} {simplifyNetlist false} {useConcatDefaultsPrefix true}
  {usefulSkew false} {verbose false} {yieldEffort none}
  ```

getSchedulingFile

`getSchedulingFile`

Returns the name of the current scheduling file used by the `skewClock` command. If no scheduling file name is set, `getSchedulingFile` returns nothing. If you have already run `skewClock`, this command reports the default file name. If you have specified or changed the scheduling file name using `setSchedulingFile`, `getSchedulingFile` returns this name.

Parameters

None

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Performing Optimization Before Clock Tree Synthesis"

getUsefulSkewMode

```
getUsefulSkewMode
     [-allNetEndPoints]
     [-ecoRoute]
     [-maxAllowedDelay]
     [-maxSkew]
     [-noBoundary]
     [-useCells]
     [-quiet]
```

Returns the following information about `setUseFulSkewMode` parameters in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software returns information for all of the `setUseFulSkewMode` mode parameters.

Parameters

`parameter_names`

Returns information for the specified parameter. You can specify one or more parameters.

See `setUsefulSkewMode` for parameter descriptions.

`-quiet`

Returns the current settings for the specified parameters in Tcl list format only.

If you specify `-quiet` without any parameters, the software returns the current settings of all `setUsefulSkewMode` parameters in Tcl list format.

Examples

- The following command returns the current setting of the `-allNetEndPoints` parameter:

  ```
  getUsefulSkewMode -allNetEndPoints
  ```

The software returns the following information:

```
-allNetEndPoints false # bool, default=false
false
```

- The following command returns the current settings for all the `setUsefulSkewMode` parameters:

  ```
  getUsefulSkewMode
  ```

  The software returns the following information:

  ```
  -allNegEndPoints false                # bool, default=false
  -ecoRoute false                       # bool, default=false
  -maxAllowedDelay 2.14748e+09          # float, default=2.14748e+09,
  min=0.000000, max=2147483647.000000
  -maxSkew false                        # bool, default=false
  -noBoundary false                     # bool, default=false
  -useCells {list_of_cell_names}
                                        # string, default=""

  {allNegEndPoints false} {ecoRoute false} {maxAllowedDelay 2.14748e+09}
  {maxSkew false} {noBoundary false} {useCells {list_of_cell_names}}
  ```

- The following command returns the current setting of the `-allNetEndPoints` parameter in Tcl list format only:

  ```
  getUsefulSkewMode -allNetEndPoints -quiet
  ```

  The software returns the following information:

  ```
  false
  ```

- The following command returns the current settings for all `setUsefulSkewMode` parameters in Tcl list format only:

  ```
  getUsefulSkewMode -quiet
  ```

  The software returns the following information:

  ```
  {allNegEndPoints false} {ecoRoute false} {maxAllowedDelay 2.14748e+09}
  {maxSkew false} {noBoundary false} {useCells {list_of_cell_names}}
  ```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Useful Skew"

insertRepeater

```
insertRepeater
     {-rule ruleFile | -template}
     [-check]
     [-checkOnly]
     [-cloneConstrainedPort {0 | 1}]
     [-embeddedBufferPortMap fileName]
     [-embeddedBufferUseMacroOnly | -embeddedBufferUseGrouteOnly]
     [-embeddedBufferVerbose filename]
     [-excNet fileName]
     [-handlePlaceBlk]
     [-keepPort]
     [-maxIter integer]
     [-minLenFix value]
     [-netMapping fileName]
     [-outfile fileName]
     [-postRoute]
     [-reportIgnoredNets fileName]
     [-ruleTolerance value]
     [-selNet fileName]
     [-useEmbeddedBuffers embedded_buffer_strength
```

Inserts buffers and inverter pairs along the routes of a net, as defined in a rule file. By default, inserts available buffers and inverters with the highest drive strengths. Supports hard blocks that have multiple input and output pins. You can set the output drive strength and input load separately for each pin.

Use this command to perform the following actions after placing the design:

- Insert buffer or inverter pairs to fix length or capacitance rules as specified in the rule file, without using any timing information.
- Insert buffers or inverter pairs in a block or at the top level of a partitioned design.

The command uses the capacitance and length rules to fix the net length before working on net capacitance for failing nets.

- For the length of a net, the command uses the maximum route length of all the source sink pairs, not the total length of the net.
- For the capacitance load, the command includes both the net capacitance and the input pin capacitance of all the sinks.

To generate a sample rule file, specify the `-template` parameter. If a rule file is already available, specify the `-rule` parameter.

The rule file contains the following information:

- For each buffer or inverter, the output drive strength must be specified in terms of the maximum wire length and maximum net capacitance it can drive. (`SetBufferDrivingStrength`)
- (Optional) For any cell, the output drive strength of all the pins can be specified in terms of the maximum wire length and maximum net capacitance it can drive. (`SetCellDrivingStrength`)
- (Optional) For any cell, the input pin load can be specified in terms of the effective wire length and pin capacitance it can contribute to a net. (`SetCellReceiverLoad`)
- For the cells apart from buffers and inverters in the library, a default output drive strength can be specified in terms of the maximum wire length and maximum net capacitance it can drive. (`SetDefaultDrivingStrength`)

  If a default drive strength is missing from the rule file, the command issues a warning and uses the lowest drive strength of any of the buffers defined in the rule file as the default drive strength.

To use embedded buffers, the following prerequisites must be met:

- The design must be congestion free.
- The embedded buffer pins and ports must be defined in Verilog.
- The embedded buffer pins and ports are available in LEF.
- You must modify the rule file to be length-based.
- You must provide a distance within which the tool can find embedded buffers.

The following example shows how to specify capacitance rules to force length-based rules to take precedence:

```
===== repeater.rule ===========
<BufferDrivingStrength>               <Name>          <microns>        <pF>
SetBufferDrivingStrength              BUFFD12          1500            1000
SetDefaultDrivingStrength                              1500            1000
================================
```

You must provide a buffer port map as follows:

```
#map file for embedded buffers
#<input>           <output>             <tieport>
macro_inst/BUF1_I   macro_inst/BUF1_O     macro_inst/BUF1_T
macro_inst/BUF2_I    macro_inst/BUF2_O     macro_inst/BUF2_T
```

Note: `macro_inst` is the full hierarchical name of the instance of the macro with embedded buffers.

Important

The `insertRepeater` command does not fix high-fanout nets with fanout greater than 1000. Therefore, before running `insertRepeater`, run the `optDesign` command or the `bufferTreeSynthesis` command to fix high-fanout nets.

Parameters

`-check` Fixes as many violations as possible and reports the nets with remaining violations.

`-cloneConstrainedPort {0 | 1}`

Specifies whether constrained ports can be cloned.
Default: `0`
Specify one of the following parameters:

- `0` Only unconstrained ports can be cloned.
- `1` All ports can be cloned.

`-checkOnly` Reports the violations without inserting repeaters.

`-embeddedBufferUseGrouteOnly`

Uses only regular buffers in the channel area, which are defined through the `-rule` option. No macro buffers will be used.

Note: You must specify `-useEmbeddedBuffers` and `-embeddedBufferPortMap` in order to use this command.

Default: The software uses both groute and macro buffers.

`-embeddedBufferUseMacroOnly`

Uses only embedded buffers within macros, which are pushed down into the macros in a hierarchical design and used at the top level for repeater insertion to correct DRVs. The software can then use the embedded buffers instead of adding buffers in the channel area. If the input terminal of the embedded buffer is tied to tie hi/lo, `insertRepeater` unties the input terminal from tie-hi/lo and uses the buffer to correct DRVs. No groute buffers will be used.

Note: You must specify `-useEmbeddedBuffers` and `-embeddedBufferPortMap` in order to use this command.

Default: The software uses both groute and macro buffers.

`-embeddedBufferVerbose` *`fileName`*

Writes messages to a file when the software cannot find an embedded buffer within the search radius of a route path.

The *`filename`* is optional. If you do no provide a *`filename`*, the tool writes messages to the Encounter log file.

`-embeddedBufferPortMap`

Specifies the file that contains buffer port mapping: input, output, and tie ports.

Note: You must specify `-useEmbeddedBuffers` in order to use this command.

`-excNet` *`fileName`* Specifies a file that contains the names of hierarchical nets to exclude from rule violation repair. If a net appears in the files specified by both `-excNet` and `-selNet`, it is excluded from rule violation repair.

`-handlePlaceBlk` Allows `insertRepeater` to support placement blockages. When this parameter is specified, if a net crosses a placement blockage, the software selects a buffer that is just strong enough to drive the blocked net length, and inserts it between the driver of the net and the blockage. The longer the blockage, the greater the strength of the buffer that is selected.

`-keepPort` Specifies a file that lists hierarchical instances whose port boundaries and polarity must not be changed when `insertRepeater` runs.

`-maxIter` *`integer`*

Specifies the maximum number iterations. when you specify this parameter, the software routes the design between iterations. If the `-postroute` parameter is specified, the software runs ECO routing with the NanoRoute router or, if `-postroute` is not specified, uses Trial Route.

An iteration can stop if all the violations are corrected or if only a few violations are being corrected in that iteration.

`-minLenFix` *`value`*

Specifies the minimum length the repeater must drive. Use this parameter to prohibit `insertRepeater` from adding buffers if the length being fixed is too small. For example, if you specify `0.3`, and the buffer length rule is 600 um, unless buffer insertion would fix a length of at least 180 um (0.3 x 600), `insertRepeater` does not insert a buffer. You must specify a value between 0 and 1.
Default: `0.2`

`-netMapping` *`fileName`*

Generates a file that maps the original net names to the nets inserted by this command.

Following is an example of the file format. In this example, `netA` is split into two nets: `FE_new_net1_netA` and `FE_new_net2_netA`.

```
netA FE_new_net1_netA
netA FE_new_net2_netA
```

If, in the next iteration, `FE_new_net1_netA` is split again, the file would contain the following:

```
FE_new_net1_netA FE_new_net1_FE_new_net1_netA
FE_new_net1_netA FE_new_net2_FE_new_net1_netA
```

Important

When you use `insertRepeater` for feedthrough insertion, you must specify this parameter if you later want to run the `hiliteFeedthroughNets` command to highlight the feedthrough paths of the nets.

`-outfile` *`fileName`*

Creates an output file that lists names of violating nets and reasons the violations were not fixed by `insertRepeater`. If you do not specify this parameter, the results are reported in the Encounter log file.

The file has two parts.

- The first part lists the names of violating nets that `insertRepeater` was not able to fix, and gives the reasons, for example:

```
Net "DTMF_INST/SPI_INST/n_2679": length not fixed
for following reason(s) placement blockage found
along net route.
```

- The second part includes the following information on the violating nets: net names, the length and capacitance rule allowed on the nets, the actual length and capacitance on the nets. It also includes the total number of violations remaining in the design. For example:

```
* info: 10.0% design error tolerance
* info: netName [maxLen maxCap] [len cap] <type of
violations>
Net DTMF_INST/SPI_INST/n_2679 [300 0.31] [349 0.11]
lenOutBound
* info: total 1 nets failed rule checking
* info: total 1 violations (1 maxLen(2.4066), 0
maxCap(0.0000))
```

`-postRoute`

Allows repeater insertion in a detail-routed design. Preserves routing so that changes are minimized. Does not do routing updates.

Note: Run ECO routing with the NanoRoute router if you specify this parameter. For more information see `ecoRoute`.

`-reportIgnoredNets` *`fileName`*

Reports the nets where no buffers were inserted and the reason the nets were ignored.

`-rule` *`ruleFile`*

Specifies a file you create that should list the buffers and inverters and the length and capacitance rules for inserting each of them. Only buffers and inserters included in the rule file are inserted.

`-selNet` *`fileName`*

Specifies a file that contains the names of hierarchical nets considered by this command. If a net appears in the files specified by both `-excNet` and `-selNet`, it is excluded from rule violation repair.

`-ruleTolerance` *`value`*

Specifies a percentage value for the length margin when reporting violations. You must specify a value between 0 and 100. For example, if you specify `20`, the software does not report violations within 20 percent of the length rule.

`-template`

Writes a template rule file you must edit to match the length and capacitance requirements of the design. The default filename is `template.rule`.

When you specify this parameter, the command does not actually insert any buffers or inverters into the netlist.

Important

If you specify the `-template` parameter, the software does not honor any other parameters for this command.

Examples

- The following is an example of a rule file:

```
# Buffer Drive Strength         Cell   Max Wirelength Total Cap
#                               Name       Microns     pF
SetBufferDrivingStrength        BUFX16      1200       2.5
SetBufferDrivingStrength        BUFX20      1500       3.0
SetInvertorDrivingStrength      INVX20      1500       3.0
# Block Output Drive Strength (Optional to customize)
SetCellDrivingStrength          alu         50         0.5
SetCellDrivingStrength          rpt_blk     2500       5.0
# Block Input Load (Optional to overwrite timing library values)
SetCellReceiverLoad             alu         50         0.05
SetCellReceiverLoad             rpt_blk     50         0.05
# Default Block Drive Strength
SetDefaultDrivingStrength                   2000       4.0
```

  - The following command specifies output drive strength rule for `BUF16`:

```
SetBufferDrivingStrength BUFX16 1200 2.5
```

This rule specifies that BUFX16 can drive at most a total wire length of 1200 µm, and a total capacitance load of 2.5 pF. The output of `insertRepeater` does not have BUFX16 drive a net with wire length greater than 1200 µm or capacitive load greater than 2.5 pF.

- The following command specifies that block a `lu` cannot drive a net with wire length greater than 50 µm and a capacitive load greater than 0.5 pF:

  ```
  SetCellDrivingStrength alu 50 0.5
  ```

- The following command specifies the input loading rule:

  ```
  SetCellReceiverLoad alu 50 0.05
  ```

  This rule specifies that the input pins of block a `lu` has an effective wirelength of 50 µm and has a pin capacitance of 0.05 pF. That is, `insertRepeater` adds the extra 50 µm wire to the wire length of the net connected to the pin.

- The following command generates a report, named `myfile`, that lists ignored nets:

  ```
  insertRepeater -reportIgnoredNets myfile
  ```

  Here is the report:

  ```
  Ignored Net Report File
  =======================

  Net Name                          Reason

  GND                               Special Net
  VDD                               Special Net
  ROM_interface/codes/n146          Analog Net
  ROM_interface/codes/n144          Net with fixed/cover wires
  ```

-

Related Topics

- Optimizing Timing in the *Encounter User Guide*
  - “Optimizing timing Using a Rule File”
  - “Default Naming Conventions”

loadFootPrint

```
loadFootPrint -infile fileName
```

Loads a footprint file to update the footprint information, which is obtained from the timing library file during design import. Loading a footprint file takes precedence over what is currently defined in the Encounter session. That is, after a footprint file is loaded for a session, only the cells in the footprint file are considered. This feature lets you run individual timing optimization commands with a limited set of cell footprints.

Parameters

`-infile` *`fileName`* Specifies the name of the footprint file.

Examples

- The following command loads the footprint file `INV.fpt`:

  ```
  loadFootPrint -infile INV.fpt
  ```

  After loading the footprint file, the Encounter software ignores the previous footprint information.

- When the `loadFootPrint` command reads the output from the `reportFootPrint` command, the `loadFootPrint` command ignores all other fields except the library name, cell name, and the footprint name. The footprint file format for the `loadFootprint` command is as follows:

  ```
  Library:library_name
  cell_name  footprint_name
  cell_name  footprint_name
  ```

  For example:

  ```
  Library:stdcellLib
  xr03d1  xr03_5
  xr03d2  xr03_5
  xr03d2  xr03_5
  ```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Cell Footprints"

optCellYield

```
optCellYield
     [-force | -keepTiming]
```

Optimizes cell yield by swapping low yield cells for higher yield cells before or after placement. Running yield optimization before placement results in more aggressive yield improvement. If you run this command without any parameters, the software does not swap cells if doing so would degrade timing or exceed local or global density.

Note: The command `setOptMode -yieldEffort` is more reliable after placement because the impact of timing, power, and area can be optimized concurrently with yield.

To optimize cell yield after routing, use the following commands:

```
setOptMode -yieldEffort high
optDesign -postRoute
```

`optCellYield` assumes that you have low and high yield cells in your library. Also, you must run `loadYieldTechFile,` either as part of design import (`loadConfig`) or as a Tcl command, before you run `optCellYield`. This reads in the cell yield coefficients (probability of failure for each cell).

Parameters

`-force`	Swaps high-yield cells for low-yield cells without regard to timing.
`-keepTiming`	Does fast swapping of cells and gives better yield than the default flow does. Uses estimates of cell timing changes during cell swapping—in some cases, the estimates differ from the actual timing changes. Honors local density constraints. The local density for yield optimization is the sum of the local density of timing optimization and a yield density margin. This margin is usually negative so that the local density for yield optimization is smaller than the local density for timing optimization.

Related Topics

- Analyzing Yield chapter in the *Encounter User Guide*

optDesign

```
optDesign
     {-preCTS | -postCTS | -postRoute}
     [[-drv] | [ -hold] | [-si]]
     [-idealClock]
     [-incr]
     [-noECORoute]
     [-selectedNets fileName]
     [-excludeNets fileName]
     [-selectedTerms fileName]
     [-outDir directoryName]
     [-prefix fileNamePrefix]
     [-useSDF]
     [-useTransitionFiles]
```

Performs timing optimization before or after the clock tree is built, or after routing and generates timing reports. You can optimize with or without useful skew.

In multi-mode, multi-corner mode, `optDesign` optimizes all analysis views concurrently. For more information, see "Performing Multi-Mode Multi-Corner Timing Analysis and Optimization" in the *Encounter User Guide*.

Performs the following optimizations:

- Corrects design rule violations.
- Reduces total negative slack.
- During the initial pass, optimizes setup time by working on the design's worst paths until the slack cannot be further improved, then, if the worst slack is not on a register-to-register path, optimizes the register-to-register paths on the second pass.
- Corrects hold time violations (optional).
- Optimizes useful skew (optional).
- Reclaims area (optional).
- Optimizes leakage power (optional).

Uses some or all of the following techniques, depending on the design stage (that is, before or after CTS, or after routing) and the specified parameters:

- Adds buffers.
- Resizes gates.
- Restructures the netlist.
- Remaps logic.
- Swaps pins.
- Deletes buffers.
- Moves instances.
- Applies useful skew.

Sets the minimum recommended and appropriate set of parameters for the following modes:

- `setAnalysisMode`
- `setTrialRouteMode`
- `setOptMode`
- `setExtractRCMode`

Tip

You can set additional parameters by specifying parameters for the appropriate `set*Mode` command or `setClockDomains` before using `optDesign`.

Before running `optDesign`, complete the following steps:

1. Set the Encounter software to handle assign nets.

   Check the Verilog netlist for Assign statements. If Assign statements exist, use one of the following procedures to enable optimization to work on those nets:

   - Specify `setDoAssign` on before loading the design data.
   - Specify `set rda_Input(assign_buffer) {1}` in the configuration file.

2. Set the default and detailed extraction scale factors by using the following commands:

   - `generateRCFactor`
   - `setRCFactor`

3. Set input transitions for the high fanout nets for delay calculation to use on high fanout networks.

Parameters

Note: The "optDesign Parameter Matrix" on page 2047 shows which parameters can be used together.

`-drv`

Corrects `max_cap` and `max_tran` violations. To correct `fanout_load` violations, specify `setOptMode -fixFanoutLoad` before you specify `optDesign`. You cannot use this parameter if you specify `-incr`.

`-excludeNets` *`fileName`*

Specifies the file that contains the hierarchical names of nets to exclude from DRV repair and fanout optimization. If a net appears in both `-excludeNets` and `-selectedNets`, it is excluded.

Note: Excluded nets, especially those with large fanout, might still be buffered to fix timing problems. Use the `dont_touch` option in the SDC constraints file to prevent `optDesign` from modifying selected nets.

`-hold`

Corrects hold violations. You cannot use this parameter if you specify `-preCTS`. You can specify `-selectedNets` or `-selectedTerms` if you use this parameter.

This parameter creates a summary file in the output directory for the design. The file is named *`prefix`*`_hold.summary`. Use the `-prefix` parameter to specify the prefix.

Important

Specifying `-hold` and `-ilm` at the same time is not supported by the software in `-postCTS` mode, but is supported in `-postRoute` mode.

To report detailed debugging information, specify the `setOptMode -hold` parameter. For more information, see setOptMode on page 2091.

`-idealClock`

Performs timing optimization in post-CTS or postroute modes using ideal clocks instead of propagated clocks. Using ideal clocks enables you to optimize timing without building a clock tree first, or ignore the existing clock tree, to close ideal timing. If this parameter is not specified, the software assumes the clock tree has been built, and optimizes timing based on propagated clocks.

`-incr`

If clock domains are honored (the default, or when `setAnalysisMode -honorClockDomains` is `true`) performs setup optimization incrementally on the worst violated paths until the worst slack cannot be optimized further, or on a path group if you specified one with `setClockDomains`.

If user-defined path-group support is honored (`setAnalysisMode -honorClockDomains` is `false`), optimizes high-effort path groups concurrently or, if no high-effort path groups are available, optimizes the worst overall slack of the low-effort path groups.

`-incr` has the following limitations:

- You cannot use it if you also specify `-si` or `-drv`.
- Without it, `optDesign` does not run when a path group other than register-to-register is selected.
- You must have already run `optDesign` on all path groups before running it with `-incr`.

`-noECORoute`

Prohibits the Encounter software from performing ECO routing with the NanoRoute® router during postroute optimization, or in useful skew optimization (`setOptMode -usefulSkew`) in post-CTS modes. During useful skew optimization, the clock net must be detail routed.

`-outDir` *`directoryName`*

Specifies the directory where the software writes timing reports generated when this command runs.

Default: `timingReports`

`-postCTS`	Performs timing optimization on a design whose clock tree has been created. By default, `-postCTS` repairs design rule violations and setup violations. If the worst negative slack does not occur on a register-to-register path, `-postCTS` performs an additional optimization pass on the register-to-register critical paths when clock domains are not set. Does not repair maximum fanout design rule violations unless you first specify `setOptMode -fixFanoutLoad`. In this mode, if you are running useful skew and you have already detail routed the clock, the software performs ECO routing using the NanoRoute router.
`-postRoute`	Performs timing optimization on a design whose routing is complete. By default, `-postRoute` repairs design rule violations and setup violations. If the worst negative slack does not occur on a register-to-register path, `-postRoute` performs an additional optimization pass on the register-to-register critical paths when clock domains are not set. Maximum fanout design rule violations are not repaired by default. To do this, first specify `setOptMode -fixFanoutLoad`.
`-preCTS`	Performs timing optimization on the placed design, before the clock tree is built. By default, `-preCTS` repairs design rule violations and setup violations. If the worst negative slack does not occur on a register-to-register path, `-preCTS` performs an additional optimization pass on the register-to-register critical paths when clock domains are not set. Maximum fanout design rule violations are not repaired by default. To do this, first specify `setOptMode -fixFanoutLoad`. **Note:** Before optimizing a design in pre-CTS mode, you must break all timing loops by disabling arcs in the constraint file. If you do not disable the arcs, the software cannot make a valid comparison of WNS between two different runs since it might not break the loops at the same point each time. To disable the arcs, use the following command: `set_disable_timing` For more information, see `set_disable_timing`.

`-prefix` *`fileNamePrefix`*

Specifies a prefix for `optDesign` report file names.

Default: *`DesignName_DesignStage`*, where *`DesignStage`* is `preCTS`, `postCTS`, or `postRoute`.

`-selectedNets` *`fileName`*

Specifies the file that contains the hierarchical names of nets to consider for design rule violation repair. Timing optimization excludes all other nets. If a net appears in both `-excludeNets` and `-selectedNets`, it is excluded.

`-selectedTerms` *`fileName`*

Specifies the file that contains the hierarchical names of instance terminals to consider for timing optimization. Only nets connected to those terminals are optimized; for example, during hold fixing, delay cells are added only on the nets driving the terminals specified in the file.

`-si`

Corrects glitch and setup violations caused by incremental delays due to coupling capacitance. You can use this parameter in `-postRoute` mode only.

`-useSDF`

Specifies that `optDesign` uses delays from SDF files for timing analysis during postroute setup or hold fixing.

- In MMMC mode, use the following command to provide an SDF file for each active view.

  ```
  read_sdf -view viewName
  ```

  For more information, see `read_sdf`.

- In non-MMMC mode, use the following commands to specify SDF files:

  ```
  setOptMode -setupSdfFile filename
  setOptMode -holdSdfFile filename
  ```

  At the end of SDF-based `optDesign` runs, the `-setupSdfFile` and `-holdSdfFile` parameters are cleared to avoid using the wrong SDF files in subsequent SDF-based `optDesign` runs.

  For more information, see `setOptMode`.

`-useTransitionFiles`

Uses the transition file specified by the `readTransitionFile` command for transition violation fixing during postroute SI optimization.

optDesign Parameter Matrix

The following table shows the compatibility of `optDesign` parameters.

optDesign	-preCTS	-postCTS	-postRoute	-drv	-hold	-idealClock	-incr	-noEcoRoute	-si
-preCTS	N/A	–	–	+	–	–	+	–	–
-postCTS	–	N/A	–	+	+	+	+	–	–
-postRoute	–	–	N/A	+	+	+	+	+	+
-drv	+	+	+	N/A	–	+	–	–	–
-hold	–	+	+	–	N/A	+	+	–	+
-idealClock	–	+	+	+	+	N/A	+	–	–
-incr	+	+	+	–	+	+	N/A	–	–
-noEcoRoute	–	–	+	–	–	–	–	N/A	–
-si	–	–	+	–	+	–	–	–	N/A

Legend

+	You can use the parameter in the row together with the parameter in the column.
–	You cannot use the parameter in the row together with the parameter in the column.
N/A	The parameters in the row and column are identical.

Examples

Pre-CTS Optimization

- To optimize timing for first time, use the following command:

```
optDesign -preCTS
```

- To repair design rule violations only, use the following command:

  ```
  optDesign -preCTS -drv
  ```

- To run incremental setup-only optimization, use the following command:

  ```
  optDesign -preCTS -incr
  ```

- To optimize using low-effort mode for design prototyping, use the following commands:

  ```
  setOptMode -effort low
  optDesign -preCTS
  ```

  The `optDesign` command performs gate resizing and global buffer insertion, but does not perform netlist restructuring or design rule violation repair.

- To run optimization with useful skew, use the following commands:

  ```
  setOptMode -usefulSkew true
  optDesign -preCTS
  ```

- To run optimization on specific path groups, use the following commands:

  ```
  clearClockDomains
  setClockDomains -fromType domainName -toType domainName
  optDesign -postCTS -incr
  ```

  For example, to run optimization on register-to-register paths, use the following commands:

  ```
  clearClockDomains
  setClockDomains -fromType register -toType register
  optDesign -postCTS -incr
  ```

  Note: If you specify a path group, `optDesign` preserves the path group setting. In the previous example, subsequent commands affect the register-to-register path group only. To set a new clock domain, specify `clearClockDomains`, followed by the definition of the clock domain you want to select.

  For example, to reset the path group to input-to-register, use the following commands after you run `optDesign`:

  ```
  clearClockDomains
  setClockDomains -fromType input -toType register
  ```

  To specify all path groups, use the following command:

  ```
  clearClockDomains
  setClockDomains -all
  ```

- To reclaim area during optimization, use the following commands:

  ```
  setOptMode -reclaimArea true
  optDesign -postCTS
  ```

- To run incremental optimization with useful skew, use the following commands:

  ```
  setOptMode -usefulSkew true
  optDesign -preCTS -incr
  ```

- To run an incremental optimization including reclaiming area, use the following commands:

```
optDesign -preCTS
setOptMode -reclaimArea true
optDesign -preCTS -incr
```

Post-CTS Optimization

- To optimize timing after the clock tree is built, use the following commands:

```
optDesign -postCTS
```

  The `optDesign` command corrects design rule violations, then setup violations.

  Note: The software does not correct fanout violations in `-postCTS` and `-postRoute` modes. To repair fanout violations, specify the following parameter before you run `optDesign`:

```
setOptMode -fixFanoutLoad true
```

- To repair setup and hold violations, use the following commands:

```
optDesign -postCTS
optDesign -postCTS -hold
```

- To repair design rule violations only, use the following command:

```
optDesign -postCTS -drv
```

- To repair hold violations only, use the following command:

```
optDesign -postCTS -hold
```

- To run optimization on specific path groups, use the following commands:

```
clearClockDomains
setClockDomains -fromType domainName -toType domainName
optDesign -postCTS
```

  For example, to run optimization on register-to-register paths, use the following commands:

```
clearClockDomains
setClockDomains -fromType register -toType register
optDesign -postCTS
```

- To reclaim area during optimization, use the following commands:

```
setOptMode -reclaimArea true
optDesign -postCTS
```

- To take advantage of useful skew when optimizing timing in post-CTS mode, use the following commands:

```
setOptMode -usefulSkew true
optDesign -postCTS
```

If you already performed detail routing on the clock tree, the Encounter software performs global and detailed ECO routing using the NanoRoute® router in post-CTS useful skew mode. If you do not want `optDesign` to do this, specify the `-noECORoute` parameter as follows:

```
setOptMode -usefulSkew true
optDesign -postCTS -noECORoute
```

If you specify `-noECORoute`, the software uses trial routing to estimate clock delays.

- To run post-CTS optimization if your design has a clock mesh, use the following commands:

  ```
  setOptMode -usefulSkew false
  optDesign -postCTS
  ```

- To run optimization on specific path groups, use the following commands:

  ```
  clearClockDomains
  setClockDomains -fromType domainName -toType domainName
  optDesign -postCTS -incr
  ```

  For example, to run incremental optimization on register-to-register paths, use the following commands:

  ```
  clearClockDomains
  setClockDomains -fromType register -toType register
  optDesign -postCTS -incr
  ```

- To optimize setup time incrementally and reduce area, use the following commands:

  ```
  setOptMode -reclaimArea true
  optDesign -postCTS -incr
  ```

- To take advantage of useful skew when optimizing timing in incremental post-CTS mode, use the following commands:

  ```
  setOptMode -usefulSkew true
  optDesign -postCTS -incr
  ```

  If you already performed detail routing on the clock tree, the software performs global and detailed ECO routing using the NanoRoute router in post-CTS useful skew mode. If you do not want the software to do this, specify the `-noECORoute` parameter as follows:

  ```
  setOptMode -usefulSkew true
  optDesign -postCTS -noECORoute -incr
  ```

  If you specify `-noECORoute`, `optDesign` uses trial route to estimate clock delays.

- To run incremental post-CTS optimization if your design has a clock mesh, use the following commands:

  ```
  setOptMode -usefulSkew false
  optDesign -postCTS
  ```

Postroute Optimization

- To optimize timing after detailed routing (repair setup violations), use the following command:

  ```
  optDesign -postRoute
  ```

  Note: The software does not correct fanout violations in `-postCTS` and `-postRoute` modes. To repair fanout violations, specify the following command before you run `optDesign`:

  ```
  setOptMode -fixFanoutLoad true
  ```

- To correct only hold violations, specify the following commands:

  ```
  optDesign -postRoute -hold
  ```

- To correct setup and hold violations, use the following commands:

  ```
  optDesign -postRoute
  optDesign -postRoute -hold
  ```

  The software repairs a hold violation only if it does not make setup slack worse than the setup target slack on a path.

- To take clock reconvergence pessimism removal (CPPR) into consideration when running timing optimization, use the `setAnalysisMode` command before you run `optDesign`. For example:

  ```
  setTimingDerate -max -clock -early 0.8 -late 1.2
  setTimingDerate -min -clock -early 0.8 -late 1.2
  setAnalysisMode -cppr true
  optDesign -postRoute
  ```

- To correct signal integrity violations, use the following command:

  ```
  optDesign -postRoute -si
  ```

- To correct hold violations, use the following commands:

```
optDesign -postRoute -hold
optDesign -postRoute -hold -si
```

  The first command repairs hold violations. The second command performs the following operations:

  - Analyzes cross coupling capacitance effects on glitch and noise
  - Back-annotates delays due to cross coupling capacitance
  - Reruns timing analysis
  - Repairs hold violations

- To input a SPEF file for use during postroute signal integrity optimization, use the following commands:

```
setSIMode -usePrevSpefFile spefFileName
optDesign -postRoute -si
```

  Note: The SPEF file you specify is used in the first loop of SI fixing only. From the second loop onwards, the software re-extracts the design.

- To run postroute setup fixing based on SDF in non-MMMC mode, use the following commands:

```
setOptMode -setupSdfFile SETUP.sdf
optDesign -postRoute -useSDF
```

- To run postroute hold fixing based on SDF in non-MMMC mode, use the following commands:

```
setOptMode -setupSdfFile SETUP.sdf -holdSdfFile HOLD.sdf
optDesign -postRoute -hold -useSDF
```

  Note: You must specify a setup SDF file when doing hold fixing.

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
- Flat Implementation Flow chapter in the *Encounter Flat Implementation Flow Guide*
  - "Place the Design and Run Pre-CTS Optimization"
  - "Run CTS and Post-CTS Optimization"
  - "Route the Design and Run Postroute Optimization"
  - "SI Closure"
- *Encounter Text Command Reference*

- clearClockDomains
- createInterfaceLogic
- setAnalysisMode
- setClockDomains
- setDelayCalMode
- setNanoRouteMode
- setOptMode
- setRCFactor
- setSIMode
- setTrialRouteMode
- setUsefulSkewMode

- Place the Design and Run Pre-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level Pre-CTS Flow in the *Encounter Hierarchical Implementation Flow Guide*
- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Partitions in the *Encounter Hierarchical Implementation Flow Guide*
- Route the Design and Run Postroute Optimization for Top-Level in the *Encounter Hierarchical Implementation Flow Guide*
- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- SI Closure in the *Encounter Flat Implementation Flow Guide*

- Partition SI Closure in the *Encounter Hierarchical Implementation Flow Guide*
- Top-Level SI Closure in the *Encounter Hierarchical Implementation Flow Guide*

optLeakagePower

```
optLeakagePower
     [-force]
     [-highEffort [-checkTNS | -noCheckTNS] | -lowEffort]
     [-postRoute | -preRoute]
```

Optimizes total leakage power of the design by swapping gates for gates with lower leakage power, without degrading timing.

Use this command after the design meets timing requirements. This command resizes only those cells that have positive slack, unless a negative target slack is specified by the user.

This command honors the effort level set by `setOptMode -effort`. You can override this effort level by specifying `-highEffort` or `-lowEffort` for this command.

When no parameters are specified, `optLeakagePower` uses the following settings:

- `-postRoute`
- `-highEffort`
- `-checkTNS`

Note: The library must contain `cell_leakage_power` statements to enable this command.

Parameters

`-checkTNS |-noCheckTNS`

Checks the total negative slack (TNS).

- In `-lowEffort` mode, maintains the worst slack.
- In `-highEffort` mode, `-checkTNS` minimizes the total negative slack difference. The leakage-power optimization process does not worsen any path that begins as worse than the target slack. Any path that starts out better than the target slack will only be worsened (in terms of slack) by leakage-power optimization up to the target slack.

The target slack is defined by `setOptMode -setupTargetSlack`. When the target slack is set to `0` (the default), this parameter does not worsen the TNS. However, to add or remove a guard band (to protect timing or reclaim more leakage power, respectively) you can set the target slack to a positive or negative value. That target slack then becomes the reference against which `optLeakagePower` operates while reclaiming leakage power.

Default: `-checkTNS`

`-force` Resizes each instance to a cell with the lowest leakage power, disregarding design rule violations and optimum timing.

`-highEffort | -lowEffort`

Determines the leakage power optimization effort level. These parameters override the `setOptMode -effort` settings.

Default: `-highEffort`

`-postRoute` Performs gate swapping within cells that have the same physical footprint (that is, the same size, same pin locations, and same LEF properties). This parameter preserves existing routing but is also recommended for the preroute stage. It is the default setting for this command.

`-preRoute` Performs gate resizing to reduce leakage. Do not use this parameter on a detail-routed design.

Important

This parameter may cause WNS, TNS, and DRV degradation even with `-checkTNS` activated, so it is recommended to first try the `-postRoute` parameter (the default value for this command) even if the database is in preroute stage.

Examples

- The following command optimizes leakage power in the design:

  ```
  optLeakagePower
  ```

- The following command forces the Encounter software to swap gates, including the gates with negative slack, up to the current negative slack:

  ```
  setOptMode -setupTargetSlack current_negative_slack
  optLeakagePower
  ```

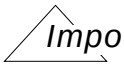
Related Topics

- Low Power Design chapter in the *Encounter User Guide*
  - "Leakage Power Optimization Techniques"

reclaimArea

```
reclaimArea
     -noDeleteBuffer
     -noDownsize
```

Creates space in the design by downsizing and deleting buffers, without worsening the slack or increasing the design rule violations.



Important

Cadence recommends using the `optDesign` command to correct timing violations during timing optimization instead of using this command. For information on using `optDesign`, see "Optimizing Timing" in the *Encounter User Guide*.

Parameters

`-noDeleteBuffer`	Prohibits the Encounter software from deleting buffers to reclaim area. *Default:* The software deletes buffers to reclaim area.
`-noDownsize`	Prohibits the Encounter software from downsizing instances to reclaim area. *Default:* The software downsizes instances to reclaim area.

Example

The following command forces the Encounter software to resize gates when there is negative slack, then reclaim area:

```
setOptMode -target_slack current_negative_slack
reclaimArea
```

reportCapViolation

```
reportCapViolation
     [-outfile fileName]
     [-selNetFile fileName]
     [-excNetFile fileName]
     [-all | -noGlobalNets]
     [-useDrcMargin]
     [-max | -min]
```

After timing analysis, reports nets that exceed the maximum capacitance constraints in the timing library and timing constraints file.

Parameters

`-all | -noGlobalNets`

Specifies the type of nets to report. The `-all` parameter reports all nets in the design. The `-noGlobalNets` parameter reports violations on all nets except for power, ground, and clock nets.
Default: `-all`

`-excNetFile` *`excNetFileName`*

Specifies the file that contains the hierarchical net names that are excluded from the capacitance violation report.

`-max | -min`

Reports only maximum capacitance violations, or only minimum capacitance violations.
Default: `-max`

`-outfile` *`fileName`*

Specifies the name of the file to which the results are written.

`-selNetFile` *`selNetFileName`*

Specifies the file that contains the hierarchical net names for the report. Only these net names are considered for inclusion in the capacitance violation report.

`-useDrcMargin`

Generates the report using the margin value that you specify using the `-drcMargin` parameter of the `setOptMode` command.

Example

The following command reports nets that exceed the maximum capacitance constraints in the report file `maxcap.report`:

```
reportCapViolation -outfile maxCap.report
```

reportCritInstance

```
reportCritInstance
     [-help]
     -outfile fileName
     [-view viewName]
     [-targetSlack slackValue]
```

Reports instances in critical paths where the slack is less than the target slack.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `reportCritInstance` parameter. For a detailed description of the command and all of its parameters, use the man command: `man reportCritinstance`.
`-outfile` *`fileName`*	Specifies the report output file. **Note:** See Examples, which show the file format.
`-targetSlack` *`slackValue`*	Reports paths with slack less than the specified *`slackValue`*. Specify the value in nanoseconds (ns). *Default:* 0
`-view` *`viewName`*	Generates the report based on the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Command Order

Use this command after using the `buildTimingGraph` command.

Examples

- The following command generates an output file named `inst.rpt`, which includes all the instance names and cell names. The instances are on the critical paths whose slack is less than 0 ns.

  ```
  reportCritInstance -outfile inst.rpt
  ```

- The following command generates an output file named `inst.rpt2`, which includes all the instance names and the cell names. The instances are on critical paths whose slack is less than `-0.3` ns. The `-targetSlack` parameter can be ignored.

  ```
  reportCritInstance -outfile inst.rpt2 -targetSlack -0.3
  ```

- The following shows the file format for `reportCritInstance`:

  ```
  Instance name:Cell name
  SUBCHIP1/XBUS_IO_REG1/XPMT12_0INV1/XPMTINV1_WFDO02XINV: IV170
  SUBCHIP1/XBUS_IO_REG1/XPMT12_0INV1/XPMTINV1_WFDO07XINV: IV170
  SUBCHIP1/XBUS_IO_REG1/XPMT12_0INV1/XPMTINV1_WFDO06XINV: IV170
  SUBCHIP1/XBUS_IO_REG1/XPMT12_0INV2/XPMTINV2_WFDO07XINV: IV170
  SUBCHIP1/XBUS_IO_REG1/XPMT12_0INV2/XPMTINV2_WFDO06XINV: IV170
  SUBCHIP1/XBUS_IO_REG1/XPMT12_0INV2/XPMTINV2_WFDO02XINV: IV17
  ```

reportCritNet

```
reportCritNet
     [-nrCritNets integer]
     [-view viewName]
     -outfile fileName
```

Generates a file containing a list of nets which have critical slack for the currently specified (setup or hold) timing analysis mode.

Parameters

`-nrCritNets` *`integer`*	Restricts the number of critical nets reported to the first *`integer`* most critical nets.
`-outfile` *`fileName`*	Specifies the name of report file.
`-view` *`viewName`*	Generates the report based on the specified analysis view only. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Command Order

Use this command after running timing optimization (using the `timeDesign` or `optDesign` commands).

Example

The following commands run timing analysis prior to clock tree synthesis, perform slack analysis, and generate a report named `critnet` that lists critical nets:

```
timeDesign -preCTS
reportCritNet -outfile critnet
```

reportCritTerm

```
reportCritTerm
     -outfile fileName
```

Performs slack analysis and writes the names of all critical terminals to the specified file. As a result, deterioration of setup slack can be avoided while the Encounter software repairs hold time violations.

Parameters

`-outfile` *`fileName`* Specifies the name of report file.

Example

- The following commands write the setup critical terminals into a file called `critTerm`:

```
setAnalysisMode -checkType setup
reportCritTerm -outfile critTerm
```

reportDontUseCells

```
reportDontUseCells
     [-outfile fileName]
     [-cell {cellName | list_of_cellNames | libName/cellName}]
```

Reports the cells prohibited from use by the timing optimization commands. This command also reports the source of the timing constraint: the timing library file, timing constraint file, or the user (by specifying the setDontUse command).

Parameters

`-cell cellName`

Specifies a cell, a list of cells, or a library and cell name.

The following formats are allowed:

- A list of cells enclosed in braces, for example:
  `-cell {buf1 and1 buf6 or5}`
- Wildcards in cell names, for example:
  `-cell *X1`
- Library and cell name, *libName/cellName*, for example:
  `-cell lib1/buf1`

`-outfile fileName`

Specifies the output file. If you do not specify this parameter, the software writes the results to the logfile.

Examples

- The following command reports cells that cannot be used for timing optimization:

  ```
  reportDontUseCells
  ```

  The software reports the following:

  ```
  ------------------------------------------------------------------
  ---                 List of Don't Use Cells                    ---
  ------------------------------------------------------------------
  ssad2       set don't use by Timing constraints
  ssmx2       set don't use by User
  ssnid2      set don't use by Timing constraints
  ssor2       set don't use by Library
  ```

- The following command writes the report to file `dontUseFile`:

  ```
  reportDontUseCells -outfile dontUseFile
  ```

- The following command reports whether `buf1`, `buf2`, `buf3`, or `buf4` is a “Don’t Use” cell.

  ```
  reportDontUseCells -cell {buf1 buf2 buf3 buf4}
  ```

  The software reports the following:

  ```
  reportDontUseCells -cell {buf1 buf2 buf3 buf4}
  dont use property at True on cell lib1/buf1, set by User
  dont use property at True on cell lib1/buf2, set by Timing constraints
  dont use property at True on cell lib1/buf3, set by Library
  dont use property at False on cell lib1/buf4
  ```

reportFanoutViolation

```
reportFanoutViolation
     [-outfile fileName]
     [-selNetFile selNetFileName]
     [-excNetFile excNetFileName]
     [-all | -noGlobalNets]
```

After timing analysis, reports all the pins that exceed the maximum fanout constraints in the timing library and timing constraints file.

Parameters

`-all | -noGlobalNets`

- `-all` reports all nets in the design.
- `-noGlobalNets` reports violations on all nets except for power, ground, and clock nets.

Default: `-all`

`-excNetFile` *`excNetFileName`*

Specifies a file that lists the names of hierarchical nets to exclude from the report.

`-outfile` *`fileName`* Specifies the report file.

`-selNetFile` *`selNetFileName`*

Specifies a file that lists the names of hierarchical nets that are considered for the report.

Example

The following command reports the pins that exceed the maximum fanout constraints in a file named `maxFanout.report`:

```
reportFanoutViolation -outfile maxFanout.report
```

reportFootPrint

```
reportFootPrint
     -outfile fileName
     [-dontTouchNUse]
     [footPrintName]
```

Creates a file that contains footprint attribute information from the timing library. Use this file to check your design for missing footprint function names and timing information. You can edit the output file and use it to load the correct footprint information.

Parameters

`-dontTouchNUse`	Prints the names of the cells that have the `DONT_TOUCH` or the `DONT_USE` attribute in the output file.
`footPrintName`	Specifies the footprint to report. If you do not specify this parameter, reports all footprint.
`-outfile` *`fileName`*	Specifies the footprint report file.

Command Order

Use this command after importing the design or after using the `loadFootPrint` command.

Examples

- The following command creates a footprint report file named `INV.rpt`:

  ```
  reportFootPrint -outfile INV.rpt
  ```

- The following output from the `reportFootPrint` command shows the format of the footprint file:

  ```
  # Format:
  # Library:<library name>
  # cell              footprint           drvResR   drvResF   functions

  # footPrint: INV nrCell: 1    Library: dp_ram128x32_4x1
  Library:stdCellLib
  inv_2               INV     1            110.983    89.427    x=(~a)
  ```

  Lines beginning with a # are comment lines, which are ignored when the footprint file is loaded. The header shows the meaning of each field:

- `Library`: `stdCellLib` (with no spaces in the string) is the library where the footprint is defined.
- `inv_2` is a footprint cell.
- `INV` is the footprint name.
- `1` is the number of functions.
- `110.983` is the average rise driving resistance.
- `89.427` is the average fall driving resistance.
- `x=(~a)` is the output function of the cell.

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Cell Footprints"

reportIgnoredNets

```
reportIgnoredNets -outfile fileName
```

Generates a report that lists the nets that are ignored by timing optimization (`optDesign`) or that might impact timing closure. You can use this command after loading a design with timing constraints.

The report lists the following types of nets:

- Nets that are ignored by timing optimization

`dontTouchNets`	Nets set as don't touch by the `set_dont_touch` SDC constraint
`externalNets`	Nets connected to I/O pad or area I/O instances
`topLevelPotentialTriStateNets`	Nets connecting tristate drivers to top-level I/Os
`clockNets`	Nets marked `clock` by the timing engine
`specialNets`	Nets that have the attribute `specialnet`
`pwrGndNets`	Nets that have the attribute `power/ground`
`analogNets`	Nets that have the attribute `analog`
`noDriverNets`	Nets that do not have a driver
`Fixed Wires/Cover Wires`	Nets that have a `Fixed/Cover` wire attribute

- Nets that might impact timing closure but are not ignored

`assignNets`	Nets that are assigned
`multiDriverNets`	Nets that have multiple drivers

Parameters

`-outfile` *`fileName`* Specifies a file name for the report.

reportPathGroupOptions

```
reportPathGroupOptions
```

Reports the values for each parameter (path group name, effort level, adjustment, target slack, critical range) for each path group. If no path groups are created, the software issues a warning.

To report the names of the path groups, use the `report_path_groups` command.

Parameters

None

Example

To create the basic path groups and report the default values of the parameters, specify the following commands:

```
createBasicPathGroups
reportPathGroupOptions
```

The software generates the following report:

```
-------------------------------------------------------------------------------
 Path_group | Effort  | Adjustment | Priority |Target Slack | Critical Range |
-------------------------------------------------------------------------------
|   clkgate |   low   |        0   |    0     |         0   |          0.2   |
|    in2reg |   low   |        0   |    0     |         0   |          0.2   |
|   reg2out |   low   |        0   |    0     |         0   |          0.2   |
|    in2out |   low   |        0   |    0     |         0   |          0.2   |
|   reg2reg |  high   |        0   |    0     |         0   |          0.2   |
-------------------------------------------------------------------------------
```

reportTranViolation

```
reportTranViolation
     [-all | -noGlobalNets]
     [-selNetFile selNetFileName]
     [-excNetFile excNetFileName]
     [-useDrcMargin]
     -outfile fileName
```

Runs timing analysis and reports pins and ports that exceed the transition constraints in the timing constraints file.

Run this command after running timing analysis (`timeDesign`) or timing optimization (`optDesign`).

Parameters

`-all | -noGlobalNets`

- `-all` reports all nets in the design.
- `-noGlobalNets` reports violations on all nets except for power, ground, and clock nets.

Default: `-all`

`-excNetFile` *`excNetFileName`*

Specifies a file that lists the names hierarchical nets that are excluded from the transition violation report.

`-outfile` *`fileName`* Specifies the report file.

`-selNetFile` *`selNetFileName`*

Specifies a file that lists the names of hierarchical nets to consider in the transition violation report.

`-useDrcMargin` Generates a report using the margin value that you specify using the `-drcMargin` parameter of the `setOptMode` command.

Example

The following command runs timing analysis and generates a `module1.tran` report file that lists the pins and ports that exceed the transition constraints:

```
reportTranViolation -outfile module1.tran
```

resetPathGroupOptions

```
resetPathGroupOptions
     [path_group_name]
     [-criticalRange]
     [-effort]
     [-slackAdjustment]
     [-targetSlack]
```

Resets the parameters of a specified path group, or of all path groups, to their default values.

- If no group name and no parameters are provided, the command resets all parameters for all path groups to their default values.
- If only the group name is provided, the command resets all parameters of that path group to their default values.
- If a path group and a parameter are provided, then only that parameter of that path group is reset to its default value.

See `setPathGroupOptions` for parameter descriptions.

Parameters

path_group_name	Specifies a path group for which to reset parameters.
parameter_names	Specifies the parameters to reset to the default value.

Examples

- To reset all parameters of all path groups to the default values, specify the following command:

  ```
  resetPathGroupOptions
  ```

- To reset all the parameters of the `reg2reg` path group to their default values, but leave the parameters of all other path groups unchanged, specify the following command:

  ```
  resetPathGroupOptions reg2reg
  ```

- To reset only the `-criticalRange` parameter of the `reg2reg` path group to its default value, and leave all the other parameters unchanged, specify the following command:

  ```
  resetPathGroupOptions reg2reg -criticalRange
  ```

resize

```
resize
     [-selNetFile selNetFileName]
     [-excNetFile excNetFileName]
     [-noFFResizing]
     [-noFootPrintChange]
     [-postIPO]
```

Upsizes and downsizes instances on paths, and fanouts of paths, that do not meet the timing constraints. An instance is upsized if it reduces the delay, or downsized to reduce the input loading. Upsizing and downsizing includes flip-flop instances, which are changed with the same cell footprint family, or with a different cell footprint name if the original functionality is preserved.

Important

Cadence recommends using the optDesign command to correct timing violations during timing optimization instead of using this command. For information on using optDesign, see "Optimizing Timing" in the *Encounter User Guide*.

Parameters

`-excNetFile` *excNetFileName*

Specifies the file that contains the hierarchical net names that are excluded from timing optimization. If a net appears in both `-excNetFile` and `-selNetFile`, it is excluded.

`-noFFResizing` Prohibits the Encounter software from resizing flip-flops.

`-noFootPrintChange` Prohibits the Encounter software from exchanging a flip-flop instance with another cell footprint family.

`-postIPO` Specifies that the design is already optimized, and performs further resizing. With this parameter, the command uses more accurate delay estimation than it would have used by default.

`-selNetFile` *selNetFileName*

Specifies the file that contains the hierarchical net (path) names for timing optimization. Only these net names are considered.

Example

The following command considers upsizing and downsizing instances on all paths and fanouts of paths, except for those in the `exc.net` file, that do not meet timing constraints:

```
resize -excNetFile exc.net
```

setBufFootPrint

```
setBufFootPrint
     [libraryName:]bufCellOrFootPrintName
```

Sets the buffer cell footprint name. This footprint name is usually assigned during design import. The Encounter software uses buffer footprint names during timing optimization and top-level buffer insertion in a partitioned design. You can specify the library that contains the footprint.

Use this command before running timing optimization.

Parameters

bufCellOrFootPrintName
: Specifies the buffer cell footprint name or a buffer cell name. When you specify a buffer cell name the Encounter software automatically finds the cell footprint name.

libraryName
: Specifies the library that contains the buffer footprint. You must insert a colon (`:`) between the library name and the buffer name when you use this parameter.

Examples

- The following command sets the buffer cell footprint name to `B_1P`:

  ```
  setBufFootPrint B_1P
  ```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Cell Footprints"

setDelayFootPrint

`setDelayFootPrint` *`delayCellOrFootprintName`*

Specifies the delay cell footprint. This footprint name is usually assigned during design import. The delay cells from the cell footprint name are used for inserting delays when you run timing optimization to fix hold-time violations.

Parameters

`delayCellOrFootprintName`

Specifies the delay cell footprint name or a delay cell name. When you specify a delay cell name, the Encounter software automatically finds the cell footprint name.

Example

The following command sets the delay footprint name to `DEL_1P`:

```
setDelayFootPrint DEL_1P
```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Cell Footprints"

setDontUse

```
setDontUse
     {cellName | libName/cellName {true | false}}
```

Sets a “don’t use” attribute on a cell, prohibiting timing optimization commands from using this cell during optimization. Optionally, you can specify a library that contains the cell. Using this command overrides the library and/or timing constraints. The “don’t use” attribute you set remains persistent through save and restore operations.

If you set the “don’t use” attribute on a cell whose “don’t use” attribute is already set, the software issues a warning message, for example:

```
encounter> setDontUse buff1 true
Setting BUFF1 as a don’t use cell

encounter> setDontUse buff1 true
Warning: Cell BUFF1 already has a dont_use attribute
```

Parameters

cellName | libName/cellName

Specifies the cell to constrain. To choose a cell in a specific library, specify *libName/cellName*. You can use wildcards to specify cell names.

true | false

Specifies whether to set or unset the “don’t use” attribute: `true` sets the “don’t use” attribute on the specified cell, and `false` unsets it.

Command Order

Use this command before you run `optDesign`.

Examples

The following example sets `buff1` and `buff2` as “don’t use” cells:

```
encounter> setDontUse buff1 true
Setting BUFF1 as a don’t use cell
encounter> setDontUse buff2 true
Setting BUFF2 as a don’t use cell
```

The following example sets `AND2` in library `libSlow` set as “don’t use” cells:

```
encounter> setDontUse libSlow/AND2 true
Setting libSlow/AND2 as a don’t use cell.
```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - “Using the Footprintless Flow”
- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*
- Run Partition CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*
- Run Top-Level CTS and Post-CTS Optimization in the *Encounter Hierarchical Implementation Flow Guide*

setInvFootPrint

`setInvFootPrint` *`invCellOrFootprintName`*

Specifies the inverter cell footprint. This footprint name is usually assigned during design import. The Encounter software uses these inverter footprint names during timing optimization and top-level inverter insertion in a partition design.

Parameters

`invCellOrFootprintName`

Specifies an inverter cell footprint name or inverter cell name. When you specify an inverter cell name, the Encounter software automatically finds the cell footprint name.

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Cell Footprints"

setLatencyFile

`setLatencyFile` *`latencyFileName`*

Specifies the latency file that the timing engine must load when the timing analysis mode is set to `-usefulSkew`.

To see the current settings for the `setLatencyFile` command, see `getLatencyFile`.

Parameters

`latencyFileName`	Specifies the latency file to load.

Command Order

Use this command after setting analysis mode to `-usefulSkew`. For more information, see `"setAnalysisMode"` on page 1808.

Example

The following example sets the latency filename to `latency1`.

```
setLatencyFile latency1
```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Performing Optimization Before Clock Tree Synthesis"

setMaxCapPerFreq

```
setMaxCapPerFreq
     [-lib libName]
     [-force]
     -freq {freq1 freq2 …}
     -cap {cap1 cap2 …}
     -pins {cell pin}
```

Adds a maximum capacitance per frequency table to the existing `max_capacitance` statement in the timing library. Timing analysis uses this table to detect violations, and `optDesign` attempts to repair violations.

The timing engine derives the frequency of an instance from the clock domain to which it is connected (the fastest clock domain, if there is more than one). If the instance is a clock instance (that is, part of the clock tree), the actual clock frequency is used. If the instance is part of the combinational logic (that is, a flip-flop, or the logic connected to the output of a flip-flop), the frequency used equals one-half of the associated clock domain frequency because a flip-flop output can only toggle at one-half the rate of the input clock. For example, a clock of 1.0 Mhz has 1,000,000 rise and fall transitions per second; however the output of a flip-flop connected to a 1.0 Mhz clock can only have 500,000 rise and fall transitions per second.

Timing analysis computes the frequency directly from the timing graph. Use the `verifyACLimitSetFreq` command to write the effective frequency onto the associated net in the database. You can then use the `dbNetFrequency` database command to check a given net's frequency value. The frequency values are stored in Hertz in the database. Modifying this value with the `dbSetNetFrequency` database command only affects `verifyACLimit`.

You can use this command any time after loading the timing library files (that is, after importing or restoring the design).

Parameters

`-cap {`*`cap1 cap2`* `…}`

Specifies a list of capacitance values that must be greater than or equal to 0. You must specify the same number of `-freq` and `-cap` values.

The software derives the maximum capacitance from a linear interpolation (or extrapolation) for a given frequency value using two frequency values in the table. For interpolation, the software uses the closest value above the frequency, and the closest value below the frequency. For extrapolation, the software uses the two closest values to the frequency.

Type: Float, specified in units of pF

`-force`

Computes the maximum capacitance from the interpolated `-freq` and `-cap` values only, and ignores any previously defined `max_capacitance` values for these pins in the library.

Default: Computes the maximum capacitance using the worst case (minimum value) between the library `max_capacitance` values and the interpolated `-cap` values.

`-freq {`*`freq1 freq2`* `…}`

Specifies a list of at least two frequency values that must be greater than or equal to 0. You must specify the same number of `-freq` and `-cap` values, and the frequency values must be specified in ascending order.

Type: Float, specified in units of Hz

`-lib` *`libName`*

Modifies only the matching cell-pins in the specified library.

Default: Modifies the matching cell-pins in all libraries

`-pins {`*`cell pin`*`}`

Adds the new capacitance constraints to the specified set of cells and pins. You can specify output or inout pins. The asterisk (`*`) wildcard character can be used in a cell or pin name. If a table associated with the pin already exists, it is replaced with the new table.

For analysis and optimization, the software uses the most restrictive (the lowest capacitance) value from either the table or the `.lib max_capacitance`. If you also specify the `-force` parameter, the `max_capacitance` value is ignored. The final value can be further modified using `.sdc set_max_capacitance` statements.

Example

Assume all output pins of a library have a `max_capacitance` of 3.5 pF from the library file.

- The following command sets the frequency dependent `max_capacitance` to 3.0 pF at a frequency of 100 Mhz, and to 1.5 pF at a frequency of 200 Mhz, for all output or inout pins for all cells in the library named `fast`:

  ```
  setMaxCapPerFreq -lib fast -freq {100e6 200e6} -cap {3.0 1.5} -pins {* *}
  ```

  As a result, for a net at 300 Mhz, an and2 cell output pin has an extrapolated `max_capacitance` value of 0 pF (which means the cell cannot be used at that frequency). An extrapolation to 50 Mhz results in a `max_capacitance` value of 3.75 pF. However, the 3.75 pF value is greater than the existing `.lib max_capacitance` limit of 3.5 pF; therefore timing analysis uses the 3.5 pF value.

- The following command applies the same constraints to all libraries with cell names matching `dff*` that have a pin named `Qbar`:

  ```
  setMaxCapPerFreq -force -freq {100e6 200e6} -cap {3.0 1.5} -pins {dff* Qbar}
  ```

  Note: The original `.lib max_capacitance` limit is ignored because the `-force` parameter is specified; therefore extrapolation to 50 Mhz results in a final `max_capacitance` constraint value of 3.75 pF.

setMaxCapPerFreqTran

```
setMaxCapPerFreqTran
     -freq {freq1 freq2 freq3…}
     -tran {tran1 tran2 tran3 …}
     -cap { {cap11 cap12 cap13…} {cap21 cap22 cap23} {cap31 cap32 cap33…} }
     -pins {cell outputpin}
     [-lib libName]
     [-force]
     [-inputpin inputpin]
```

Sets the maximum capacitance of a pin in the design. Considers both the input transition time and the output frequency. If the maximum capacitance exceeds this value, the reportCapViolation command displays a capacitance violation.

You can use this command any time after loading the timing library files (that is, after importing or restoring the design).

Parameters

`-cap {{cap11 cap12 cap13…} {cap21 cap22 cap23…} {cap31 cap32 cap33…}}`

Specifies a table of capacitance values with the maximum output pin capacitances associated with `-freq` and `-tran` values. The total number of `-cap` values must be the total number `-freq` values times the total number of `-tran` values. The `-freq` and `-tran` values must be specified before `-cap`.

The capacitance values are floats in units of pico farad and all values must be greater than or equal to zero. The capacitance table uses frequency as the first index and the input transition as the second index. For example, `cap11`, `cap12`, and `cap13` are for `freq1`, while `c12`, `c22`, and `c32` correspond to the input transition `tran2`.

The maximum capacitance is derived from a linear interpolation (or extrapolation) for a given frequency value and a transition value using two frequency values and two transition values n the table (closest above and below for interpolation, the two closest for extrapolation.

Type: Float, specified in units of pF

`-force`

Computes the maximum capacitance from the interpolated `-freq, -tran,` and `-cap` values only, and ignores any previously defined `max_capacitance` values for these pins in the library.

Default: (`-force` not specified) Computes the maximum capacitance using the worst case (minimum value) between the library `max_capacitance` values and the interpolated `-cap` values.

`-freq {`*`freq1 freq2 freq3`*`…}`

Specifies a list of at least two frequency values with the toggle frequencies at the output pin. Frequency values must be increasing, and the total number of frequency values cannot exceed 255.

Type: Float, specified in units of Hz. All values must be equal to or greater than `0`.

`-inputpin` *`inputpin`*

Specifies the input pin name corresponding to the input transition values in `-tran`. If –*`inputpin`* is not given, `-tran` values correspond to all input pins.

`-lib` *`libName`*

Specifies the library name from inside a `.lib` file to update the cell-pins.

Default: If a library name is not specified, modifies the matching cell-pins in all libraries

`-pins {`*`cell`* `output`*`pin`*`}`

Adds the new capacitance constraints to the specified set of cells and pins. You can specify output or inout pins; input pins are ignored. The asterisk (`*`) wildcard character can be used in a cell or pin name.

If there is already a table associated with the same input-output pair (input pins are specified using `-inputpin`), it is replaced with the new constraints; otherwise, it is added to the output maximum capacitance table list.

For analysis and optimization, the software uses the most restrictive (the lowest capacitance) value from either the table or the `.lib max_capacitance`. If you also specify the `-force` parameter, the `max_capacitance` value is ignored. The final value can be further modified using `.sdc set_max_capacitance` statements.

`-tran {`*`tran1 tran2 tran3`* `…}`

Specifies a list of at least two transition values with the input transition times. Transition times must be increasing in value, and the total number of transition times cannot exceed 255.

Type: Float, specified in units of nanoseconds, and all values must be equal to or greater than `0`.

Example

The following command establishes a maximum capacitance table for all pins on all cells in all libraries.

```
setMaxCapPerFreqTran -force -pins {* *} -freq {1e8 1.25e8 1.5e8} -tran {450 475
500} -cap {{0.022 0.017 0.012} {0.02 0.014 0.01} {0.018 0.013 0.008}}
```

Here is the table:

```
          450         475         500
1e8       0.022       0.017       0.012
1.25e8    0.02        0.014       0.01
1.5e8     0.018       0.013       0.008
```

setMaxTranPerFreq

```
setMaxTranPerFreq
     [-lib libName]
     [-force]
     -freq {freq1 freq2 ...}
     -tran {tran1 tran2 ...}
     -pins {cell pin}
```

Adds a maximum transition per frequency table to the existing `max_transition` statement in the timing library. Timing analysis uses this table to detect violations, and `optDesign` attempts to repair violations.

Use this command any time after loading the timing library files (that is, after importing or restoring the design).

The timing engine derives the frequency of an instance from the clock domain to which it is connected (the fastest clock domain, if there is more than one). If the instance is a clock instance (that is, part of the clock tree), the actual clock frequency is used. If the instance is part of the combinational logic (that is, a flip-flop, or the logic connected to the output of a flip-flop), the frequency used equals one-half of the associated clock domain frequency because a flip-flop output can only toggle at one-half the rate of the input clock. For example, a clock of 1.0 Mhz has 1,000,000 rise and fall transitions per second; however the output of a flip-flop connected to a 1.0 Mhz clock can only have 500,000 rise and fall transitions per second.

Timing analysis computes the frequency directly from the timing graph. Use the `verifyACLimitSetFreq` command to write the effective frequency onto the associated net in the database. Use the `dbNetFrequency` database command to check any given net's frequency value. The frequency values are stored in Hertz in the database. Modifying this value with the `dbSetNetFrequency` database command only affects `verifyACLimit`.

Parameters

`-force`
: Computes the maximum transition from the interpolated `-freq` and `-tran` values only and ignores any previously defined `max_transition` values for these pins in the library.

  Default: Uses the worst case (minimum value) between the `.lib` `max_transition` values and the interpolated `-tran` values.

`-frequency {`*`freq1 freq2`* `…}`

Specifies a list of at least two frequency values that must be greater than or equal to 0. You must specify the same number of `-freq` and `-tran` values, and the frequency values must be specified in ascending order.

Type: Float, specified in units of Hz

`-lib` *`libName`*

Modifies only the matching cell-pins in the specified library.

Default: Modifies the matching cell-pins in all libraries.

`-pins {`*`cell pin`*`}`

Adds the new transition constraints to the specified set of cells and pins. You can specify input, output, or inout pins. The asterisk (*) wildcard character can be used in a cell or pin name. If a table associated with the pin already exists, it is replaced with the new table.

For analysis and optimization, the software uses the most restrictive (the lowest transition) value from either the table or the `.lib max_transition`. If you also specify the -force parameter, the `max_transition` value is ignored. The final value can be further modified using `.sdc set_max_transition` statements.

`-tran {`*`tran1 tran2`* `…}`

Specifies a list of transition values that must be greater than or equal to 0. You must specify the same number of `-freq` and `-tran` values.

The software derives the maximum transition limit from a linear interpolation (or extrapolation) for a given frequency value using two frequency values in the table. For interpolation, the software uses the closest value above the frequency, and the closest value below the frequency. For extrapolation, the software uses the two closest values to the frequency.

Type: Float, specified in units of ns

Examples

Assume all output pins of a library have a `max_transition` value of 0.9 ns from the library file.

- The following command sets the frequency dependent `max_transition` to 1.0 ns at a frequency of 100 Mhz, and to 0.7 ns at a frequency of 200 Mhz, for pins matching `in*` of cells matching `*` (that is, all cells) in the library named `slow`:

  ```
  setMaxTranPerFreq -lib slow -freq {100e6 200e6} -tran {1.0 0.7} -pins {* in*}
  ```

  As a result, at 300 Mhz, an `and2` cell input pin named `in1` has an extrapolated `max_transition` value of 0.4ns. An extrapolation to 50 Mhz results in a `max_transition` value of 1.15 ns. However, the 1.15 ns value is greater than the existing `.lib` `max_transition` value of 0.9 ns; therefore timing analysis uses the 0.9 value.

- The following command applies the same constraints to all libraries with cell names matching `dff*` that have a pin named `clk`:

  ```
  setMaxTranPerFreq -force -freq {100e6 200e6} -tran {1.0 0.7} -pins {dff* clk}
  ```

  Note: The original `.lib` `max_transition` value is ignored because the `-force` parameter is specified; therefore an extrapolation to 50 Mhz results in a final `max_transition` limit of 1.15 ns.

setOptMode

```
setOptMode
     [-help]
     [-reset]
     [-addInst {true | false}]
     [-addInstancePrefix prefix]
     [-addNetPrefix prefix]
     [-addPortAsNeeded {true | false}]
     [-allowOnlyCellSwapping {true | false}]
     [-bufferAssignNets {true | false}]
     [-clkGateAware {true | false}]
     [-congOpt {true | false}]
     [-considerNonActivePathGroup {true | false}]
     [-criticalRange value]
     [-critPathCellYield {true | false}]
     [-deleteInst {true | false}]
     [-downsizeInst {true | false}]
     [-drcMargin margin]
     [-dynamicPowerEffort {none | low | high}]
     [-effort {low | high}]
     [-fixDRC {true | false}]
     [-fixFanoutLoad {true | false}]
     [-fixHoldAllowSetupTnsDegrade {true | false}]
     [-holdFixingEffort {low | high}]
     [-holdSdfFile filename]
     [-holdTargetSlack slack]
     [-honorFence {true | false}]
     [-ignorePathGroupsForHold {groupA groupB ...}]
     [-keepPort keepPortHinstFile]
     [-leakagePowerEffort {none | low | high}]
     [-maxDensity density]
     [-moveInst {true | false}]
     [-optimizeConstantNet {true | false}]
     [-optimizeFF {true | false}]
     [-optimizeNetsAcrossDiffVoltPDs {true | false}]
     [-postRouteAllowOverlap {true | false}]
     [-postRouteSiAware {true | false}]
     [-preserveAssertions {true | false}]
     [-preserveModuleFunction {true | false}]
     [-reclaimArea {true |false}]
     [-resizeShifterAndIsoInsts {true | false}]
     [-restruct {true | false}]
     [-setupSdfFile filename]
     [-setupTargetSlack setupTargetSlack]
     [-simplifyNetlist {true | false}]
     [-sizeOnlyFile filename]
     [-swapPin {true | false}]
     [-unfixClkInstForOpt {true | false}]
     [-useConcatDefaultsPrefix {true | false}]
     [-usefulSkew {true | false}]
```

```
[-verbose {true | false}]
[-virtualPartition {true | false}]
[-yieldEffort {none | low | high}]
```

Sets global parameters for timing optimization. Parameters that you specify with `setOptMode` are then used automatically when you run timing optimization by invoking the `optDesign` command or by using the Timing – Optimization form.

To see the current settings for the `setOptMode` command, see `getOptMode`.

Parameters

`-addInst {true | false}`

Controls whether timing optimization can add instances.
Default: `true`

`-addInstancePrefix` *`prefix`*

Specifies a prefix string to add to instances created by timing optimization. For example, if a default prefix is `OFC_000234`, when you use the command `setOptMode -prefix POSTROUTE`, the combined prefix added to the instance name is `POSTROUTE_OFC_00234`.

Default: `""` (empty string)

Note: To ignore the default prefix, use the following parameter: `-useConcatDefaultsPrefix false`. For more information see `-useConcatDefaultsPrefix`.

`-addNetPrefix` *`prefix`*

Specifies a prefix string to add when timing optimization creates nets. For example, if a default prefix is `OFN_000234`, when you use the command `setOptMode -addInstancePrefix POSTROUTE`, the combined prefix added to the net name is `POSTROUTE_OFN_00234`.
Default: `""` (empty string)

Note: To ignore the default prefix, use the following parameter: `-useConcatDefaultsPrefix false`. For more information see `-useConcatDefaultsPrefix`.

`-addPortAsNeeded {true | false}`

Controls whether timing optimization adds ports to the lower-level hierarchies of the design. Prohibiting the software from adding ports (when this parameter is set to `false`) might not be optimal because buffers are not allowed to be added exactly where needed; however, this preserves hierarchical boundaries of the design so that generated test vectors are still valid, and module-by-module verification is possible.
Default: `true`

`-allowOnlyCellSwapping {true | false}`

Forces the post-route setup optimization to perform only the Vth swapping for timing improvements. This option impacts only `optDesign -postRoute` and has an effect only for the multi-Vth libraries design.
Default: `false`

Note: Use this option for reoptimizing the timing in specific cases where routing should remain unchanged.

`-bufferAssignNets {true | false}`

Controls whether timing optimization replaces all Assign nets with buffers. All assigns on nets are replaced by smallest buffer available. The `setDoAssign` command must be set to `on` (the default) before you use this parameter.
Default: `false`

`-clkGateAware {true | false}`

Specifies that optimization is aware of clock gate cells in the design. Based on the estimated clock delays downstream from the clock gating-enabled pins, `optDesign` temporarily adds tight assertion on each clock gate so that it can run on those paths during pre-CTS. The assertions are removed when `optDesign` ends before generating the final summary report.

This parameter impacts only the pre-CTS `optDesign`.

Default: `false`

Note: The `setPlaceMode` command also has a `-clkGateAware` parameter. For best results, if you set `setOptMode -clkGateAware` to `true`, set `setPlaceMode -clkGateAware` to `true` also.

`-congOpt {true | false}`

Reduces local congestion during timing optimization. This parameter applies to critical path optimization only.
Default: `false`

`-considerNonActivePathGroup {true | false}`

Ensures that, when timing optimization is working on a specific path group, it does not degrade the other non-active path group. This parameter is used to avoid problems when logic is shared between path groups. It is not recommended for default usage on all designs or in all generic scripts because it might limit timing optimization on some path groups. This parameter impacts only `optDesign` and is used each time that `optDesign` works on a specific path group.
Default: `false`

Note: This parameter supports only the main path groups: reg2reg, in2reg, reg2out, and in2out. Other non-active path groups may be degraded.

`-criticalRange` *`value`*

Optimizes paths whose negative slack falls within a calculated range. Calculating a critical range allows the software to continue optimization on paths other than the most critical path.

You can use this parameter during incremental optimization before or after clock-tree synthesis and after routing.

The *`value`* must be a decimal value between `0.0` and `1.0`.

- A *`value`* of `0.0` allows optimization on the critical path only.
- A *`value`* of 1.0 allows optimization on any path with negative slack. Setting a *`value`* of `1.0` is likely to increase optimization runtime significantly.

The critical range is calculated as follows:

`Critical Range = Worst Slack x (1.0 -` *`value`*`)`

For example, if the worst slack is `-1.0` and the *`value`* is `0.3`, paths with slack ranging from `-1.0` to `-0.7` are in the critical range, so they are optimized. The range is calculated dynamically—when the worst slack changes, the critical range changes as well. In the example above, if the worst slack changes to `-0.5`, the range is recalculated as `-0.5` to `-0.35`.

Default: `0.2`

`-critPathCellYield {true | false}`

Swaps cells on critical paths for higher-yielding cells before swapping cells on other paths. Swapping these cells first substantially reduces the chance of timing failure on the die.
Default: `false`

`-deleteInst {true | false}`

Controls whether timing optimization can delete instances.
Default: `true`

`-downsizeInst {true | false}`

Controls whether timing optimization can downsize instances.
Default: `true`

`-drcMargin` *`margin`*

Scales the `maxCap` and `maxTran` constraints according to the *`margin`* decimal value specified. For example, specifying a value of `0.2` for this parameter tightens the margin (makes it more pessimistic) by 20 percent during optimization and specifying a value of `-0.2` relaxes the margin (makes it more optimistic) by 20 percent. Introducing a pessimistic margin improves correlation between the number of DRC violations before and after routing.
Default: `0`

`-dynamicPowerEffort {none | low | high}`

Specifies the effort level for optimizing the power consumed by cells when they change state.

Default: `none`

Specify one of the following parameters:

`low` Use the low-effort mode when

- Dynamic power is not the highest priority
- Power optimization is done only in the postroute stage
- Timing closure remains the first priority and the worst negative slack achieved is same as in the `none` effort mode
- The CPU run-time impact is less than eight percent.

`high` Use the high-effort mode when

- Dynamic power reduction is very critical
- Power optimization is done in the entire `optDesign` stage
- Timing closure remains the first priority
- The CPU run-time impact is less than 20 percent

`none` Specifies no power optimization.

`-effort {low | high}`

Specifies the effort level for timing optimization. See "Low and High Effort Defaults" on page 2109 for a table that lists the default effort value and availability of parameters in low and high effort mode.
Default: `-high`

Specify one of the following parameters:

`high`	Specifies high effort level. Use this level to reach timing closure for challenging designs. This level activates all the physical synthesis optimization transforms.
`low`	Specifies low effort level. Use this level for design prototyping. This level triggers global resizing and buffering in order to obtain a good timing result in the fastest run time.

`-fixDRC {true | false}`

Repairs all nets that are constrained for maximum capacitance violations and maximum transition violations. The capacitance and transition constraint values are specified in the timing constraint file or in the library. If you specify `false` for this parameter, the software repairs only the paths with timing violations.
Default: `true`

`-fixFanoutLoad {true | false}`

Forces timing optimization to correct fanout load violations, which can be maximum load or maximum fanout count violations, depending on the libraries.
Default: `false`

`-fixHoldAllowSetupTnsDegrade {true | false}`

Controls whether the software allows total negative slack to degrade during hold fixing operations.

By default, the software allows the slack to degrade in order to fix more hold violations. To override the default behavior and ensure that the setup total negative slack is maintained during hold fixing, specify `false` for this parameter.
Default: `true`

`-help`

Outputs a brief description that includes type and default information for each `setOptMode` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man setOptMode`.

`-holdFixingEffort {low | high}]`

Specifies the effort level for repairing hold violations during hold-fixing step.

Note: In the recent Encounter version, hold fixing has been enhanced, and this implies that the `low` and `high` fixing effort have the same behavior.

`-holdSdfFile` *`filename`*

Specifies the SDF file to use for hold timing during postroute timing optimization in non-MMMC mode. The command `optDesign -useSDF` performs the optimization and then resets this parameter so the same SDF file is not used for subsequent runs.
Default: `""` (empty string)

Note: You must specify a setup SDF file when doing hold fixing.

Use this parameter in non-MMMC mode only. In MMMC mode, use the following command to provide SDF files for each active view:

```
read_sdf -view viewName
```

For more information, see `read sdf`.

`-holdTargetSlack` *`holdTargetSlack`*

Specifies a target slack value in nanoseconds to use for hold analysis only. During setup violation repair, the setup target slack is *`setupTargetSlack`* and the hold target slack value is `0`. Generally, you use both `-setupTargetSlack` and `-holdTargetSlack` for hold analysis. If you specify `-holdTargetSlack` only, the `-setupTargetSlack` value is `0`.
Default: `0`

`-honorFence {true | false}`

Specifies that the timing optimization takes fences or region constraints into account. This option is honored by each `optDesign` step. If you optimize design with the parameter set to `true`, the placement is legal.

This feature is useful for hierarchical designs.

Because this parameter adds constraints to timing optimization, enable it only when needed.

Default: `false`

Note: `optDesign` will not honor very small fences when this parameter is set to `true` if it would cause fence utilization to be greater than 100 percent.

`-ignorePathGroupsForHold {`*`groupA groupB ...`*`}`

Specifies which clock domains or path groups should be excluded from hold timing optimization. This option impacts only `optDesign -postCTS`/`-postRoute -hold`.

Default: `""` (empty string)

Note: To reset this option, apply an empty list.

Note: If a violated path is in two path groups and only one of those paths is specified in the `-ignorePathGroupsForHold` parameter, this path is still fixed during hold fixing.

`-keepPort` *`keepPortHinstFile`*

Specifies a file that contains the hierarchical instances whose port boundaries cannot be changed. The specified ports are not removed or duplicated and their polarity is unchanged.
Default: `""` (empty string)

`-leakagePowerEffort {none | low | high}`

Enables timing optimization to run in a power leakage-aware mode. In this mode, the software reduces the total amount of leakage power in addition to optimizing timing and area, and minimizes the number of design rule violations. During library set up, you must load all the Vth libraries. The software sorts the cells—you do not need to declare the high Vth library.
Default: `none`

If you specify `low` or `high` for this parameter, you do not need to run the `optLeakagePower` command because `optDesign` runs it automatically.

Specify one of the following parameters:

`none`	Disables power leakage-aware mode for `optDesign`.
`low`	Reduces leakage power only in the postroute stage. The software does not insert high-leakage cells during hold fixing.
`high`	Reduces leakage in pre-route and postroute stages. The software does not insert high leakage cells during hold fixing. This mode is more aggressive than `low` effort mode, and might affect `optDesign` run time and timing closure.

`-maxDensity` *`density`*

Specifies the maximum value for density (area utilization). `optDesign` does not grow the netlist above this value. When reaching a density close to the value specified by this parameter, `optDesign` has limited optimization capability.
Default: `0.95`

`-moveInst {true | false}`

Allows cells in the critical path to move when the software repairs setup violations or optimizes timing along the critical path, if allowing them to move improves timing.
Default: `true`

`-optimizeConstantNet {true | false}`

Controls whether timing optimization optimizes constant nets.
Default: `true`

`-optimizeFF {true | false}`

Controls whether timing optimization upsizes or downsizes flip-flops.
Default: `true`

`-optimizeNetsAcrossDiffVoltPDs {true | false}`

Controls whether timing optimization optimizes nets crossing power domains with different voltages.
Default: `true`

`-postRouteAllowOverlap {true | false}`

Limits the placement and routing interruptions caused by post-route optimization as there should be no movement on the existing cells, but might be over-constraining when there are too many timing violations to be fixed. It can be helpful for very late stage optimization to close on a few last violations and also for high utilization designs.

When set to `true`, `optDesign` performs a fully legalized placement.

When set to `false`, it is forces the post-route setup and SI timing optimization engines to find a legal location for any inserted cell or resized cell.
Default: `true`

`-postRouteSiAware {true | false}`

Controls the SI effects while fixing the base timing during post-route optimization. When set to `true`, the software reduces the SI violations, which results in faster timing closure at the signoff stage. This parameter takes effect when the `optDesign -postRoute` command is run.

Note: Although this option increases the turnaround time of the `optDesign -postRoute` command, it reduces the time taken by the `optDesign -postRoute -si` command. As a result, the run time of the complete flow is impacted marginally.

Default: `false`

`-preserveAssertions {true | false}`

Controls whether timing optimization can remove an instance that has a timing constraint applied on its pin and create a new timing constraint so that the timing graph is still valid.

With the default setting (`false`), when a new timing constraint is created, the `saveDesign` command creates a new timing constraint file and points to it in the `.conf` file.

Note: Cadence recommends that you use the default setting because it gives more flexibility to timing optimization algorithms.

This parameter affects the following commands:

- `deleteBufferTree`
- `optDesign`
- `placeDesign`
- `reclaimArea`

Default: `false`

`-preserveModuleFunction {true | false}`

Determines whether to preserve logical functions at hierarchical module ports. For example, you can prevent the Encounter software from splitting an inverter pair across a port, and thereby inverting the logical function at the port.
Default: `true` on constrained ports; `false` for all other module pins

`-reclaimArea {true | false}`

Controls whether timing optimization creates additional space by downsizing gates or deleting buffers, while maintaining worst slack and total negative slack.
Default: `false`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setOptMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-resizeShifterAndIsoInsts {true | false}`

Controls whether timing optimization resizes shifters and isolation instances. In default mode (`false`), shifters and isolations are considered “don’t-touch” instances and are not changed during timing optimization.
Default: `false`

`-restruct {true | false}`

Controls whether timing optimization swaps pins and restructures the netlist.
Default: true

Note: This parameter is not enabled when `-effort` is set to `low`.

`-setupSdfFile` *`filename`*

Specifies the SDF file to use for setup timing during postroute optimization in non-MMMC mode. The command `optDesign -useSDF` performs the optimization and then resets this parameter so the same SDF file is not used for subsequent runs.

Default: `""` (empty string)

Use this parameter in non-MMMC mode only. In MMMC mode, use the following command to provide SDF files for each active view:

```
read_sdf -view viewName
```

For more information, see `read sdf`.

`-setupTargetSlack` *`setupTargetSlack`*

Specifies a target slack value in nanoseconds for setup analysis. During setup violation repair, the setup target slack is *`setupTargetSlack`* and the hold target slack is set to `0`. Generally, you use both `-setupTargetSlack` and `-holdTargetSlack` for hold analysis. If you specify `-holdTargetSlack` only, the `-setupTargetSlack` value is `0`.
Default: `0`

`-simplifyNetlist {true | false}`

Controls whether timing optimization simplifies the netlist. The software recovers area, decreases congestion, and improves runtime by simplifying the netlist in the following ways:

- Removing dangling output instances
- Propagating constants
- Removing unobservable logic
- Remapping useless logic

The software respects the `set_dont_touch` constraint, so it does not remove the constrained flip-flops, and does not touch non-uniquified modules.
Default: `false`

Note: This parameter is not enabled when `-effort` is set to `low`.

If a netlist contains spare cells, ensure that you define them using the `specifySpareGate` command else they might be removed by the `-simplifyNetlist` parameter.

`-sizeOnlyFile` *`filename`*

Specifies a file that lists instances that may be resized but not deleted during timing optimization. The `placeDesign` command also honors this file.

In the file

- The software considers only the first instance name on each line.
- The software ignores white spaces before or after instance names.
- Wildcards are supported in instance names.
- If a hierarchical instance is specified, the `-sizeOnlyFile` attribute is applied to all the instances that belong to it.

A section of a size-only file is shown below:

```
instName_1
instName_4
instName_6
instName_9
instName_2
instName_3
instName_8
instName_0
```

Default: `""` (empty string)

`-swapPin {true | false}`

Controls whether timing optimization allows pin swapping.
Default: `true`

This parameter is not enabled when `-effort` is set to `low`.

`-unfixClkInstForOpt {true | false}`

Controls whether the software is allowed to resize fixed sequential elements and to move them during placement legalization. `optDesign` temporarily removes the fixed attribute of each sequential element at the beginning of this step and sets it back to its original value at the end. This option does not remove the routed wires of the clock tree. It minimizes the movement of sequential elements as much as possible. This option impacts only `optDesign -postCts` and `optDesign -postRoute`.

- When `true`, the software is allowed to resize and move fixed registers.
- When `false`, the software is not allowed to resize or move fixed registers.

Default: `false`

Note: If you specify true for this parameter, Cadence recommends you use the `routeDesign` command for routing your design.

`-useConcatDefaultsPrefix {true | false}`

Controls whether to combine the default prefix for cells and nets added by timing optimization with the prefix specified by `-addNetPrefix` and `-addInstancePrefix`. By default, the software combines the specified prefixes with the default prefix. Specify `false` for this parameter to ignore the default prefix and use only the prefix specified by `-addNetPrefix` and `-addInstancePrefix`.

For example, if the default prefix is `OFN_000234` and you type the following command:

```
setOptMode -useConcatDefaultsPrefix true
-addNetPrefix POSTROUTE_N -addInstancePrefix
POSTROUTE_C
```

The software uses the following prefixes:

- For cells added by optimization: `POSTROUTE_C_OFN_00234`
- For nets added by optimization: `POSTROUTE_N_OFN_00234`

If if the default prefix is `OFN_000234` and you type the following command:

```
setOptMode -useConcatDefaultsPrefix false
-addNetPrefix POSTROUTE_N -addInstancePrefix
POSTROUTE_C
```

The software uses the following prefixes:

- For cells added by optimization: `POSTROUTE_C`
- For nets added by optimization: `POSTROUTE_N`

Note: If you specify `false` for this parameter, you must also specify `-addInstancePrefix` and `-addNetPrefix`. In addition, you must specify a different prefix for instances and nets.

Default: `true`

`-usefulSkew {true | false}`

Controls whether timing optimization runs the `skewClock` command.

This parameter produces scheduling and latency files before CTS runs, or adds buffers and inverters after CTS, just as `skewClock` does. This parameter sets `setAnalysisMode -usefulSkew true`, so timing analysis can read the latency file. Useful skew optimization is run iteratively within `optDesign`, so you do not need to specify this parameter more than once.
Default: `false`

Note: This parameter is not enabled when `-effort` is set to `low`.

Note: This parameter allows clock skew optimization at the post-route stage.

`-verbose {true | false}`

Controls the amount of information timing optimization outputs to the log file. Specify `true` to output additional information to use to debug design issues.
Default: `false`

`-virtualPartition {true | false}`

Ensures that timing optimization applies to ART-based flow. In the ART-based flow, it partially activates the design at the top-level and at the interface path of partition blocks so that there is a reduction in memory usage and run time if the design is large.

This parameter marks the core logic inside the partition as `dont_touch`, so the logic is not optimized.
Default: `false`

Note: This parameter requires partition definition information and applies to the post-route mode only.

`-yieldEffort {none | low | high}`

Specifies the level of effort for performing yield optimization. This parameter applies during preCTS, postCTS, and postroute modes. It does not apply during incremental optimization.

- Low-effort yield optimization improves the yield while maintaining the timing. This effort is recommended for timing-critical designs.
- High-effort yield optimization achieves the maximum yield, but could lead to a small degradation in timing. If that occurs, an incremental optimization can be performed to recover the timing.

Default: none

Low and High Effort Defaults

The following table shows the default values and availability of parameters in low and high effort modes.

Parameter	Low Effort Default	High Effort Default
-addInst	true	true
-addPortAsNeeded	true	true
-congOpt	false	false
-criticalRange	Disabled	0.2
-critPathCellYield	false	false
-considerNonActivePathGroup	false	false
-optimizeNetAcrossDiffVoltPDs	true	true
-preserveAssertions	false	false
-preserveModuleFunction	false	false
-deleteInst	true	true
-downsizeInst	true	true
-drcMargin	0.0	0.0
-fixDRC	true	true
-fixFanoutLoad	false	false

Parameter	Low Effort Default	High Effort Default
-holdTargetSlack	0	0
-keepPort	Off	Off
-maxDensity	0.95	0.95
-moveInst	false	true
-optimizeConstantNet	true	true
-optimizeFF	true	true
-reclaimArea	false	true
-restruct	false	true
-setupTargetSlack	0.0	0.0
-simplifyNetlist	false	false
-swapPin	false	true
-usefulSkew	false	false

Examples

- The following command resets the `-yieldEffort` parameter to its default value:

```
setOptMode -reset -yieldEffort
```

- The following command resets all `setOptMode` parameters to their default values:

```
setOptMode -reset
```

- The following commands specify target slack values for setup and hold times and optimize timing in the design based on those values:

```
setOptMode -setupTargetSlack 0.1 -holdTargetSlack 0.15
optDesign -postCTS
```

- The following commands optimize leakage power in addition to optimizing timing and area, and fixing design rule violations:

```
setOptMode -leakagePowerEffort high
optDesign -preCts
optDesign -postCts
optDesign -postRoute
optDesign -postRoute -hold
```

- The following commands allow the software to move clock instances during post-CTS timing optimization, and then run timing optimization:

```
specifyClockTree -clkfile clock.stsch
timeDesign -postCts
```

```
setOptMode -unfixClkInstForOpt true
optDesgin -postCts
```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Default Naming Conventions"
- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Run CTS and Post-CTS Optimization in the *Encounter Flat Implementation Flow Guide*

setPathGroupOptions

```
setPathGroupOptions
     groupName
     [-criticalRange value]
     [-effortLevel {low | high}]
     [-slackAdjustment value]
     [-slackAdjustmentPriority integer]
     [-targetSlack float]
```

Specifies options to guide timing optimization and timing analysis when path groups are used.

To reset parameters to their default value, use `resetPathGroupOptions`.

Parameters

`-criticalRange value`

Specifies the critical range for timing optimization (`optDesign`) while optimizing a specific path group. This parameter is honored only for path groups with `-effortLevel` set to `high`.

Specifying a critical range for a specific path group is the same as specifying a critical range for `optDesign`—except that it is applied to a specific path group only. If the critical range for a specific path group differs from the critical range for `optDesign`, the software applies the higher critical range value to the specific path group.

For more information, see `setOptMode -criticalRange`.

`-effortLevel {low | high}`

Specifies the effort applied to close timing on a specific path group. This parameter is honored by timing optimization (`optDesign`).

Select one of the following options:

`low` Specifies that this path group is optimized together with other path groups for best overall slack. `low` is the default effort value for all user-created path groups.

Tip

Cadence recommends you specify low effort for all path groups except that ones that the tool should focus on most strongly.

`high` Specifies that this path group is optimized separately from other path groups in order to achieve the best possible WNS for this path group.

`groupName` Specifies the path group for which to set options.

`-slackAdjustment` *`value`*

Specifies a positive or negative value used to modify path constraints on all paths that are part of a specific path group. The value is in nanoseconds.

- To relax the path constraints computed by the timing engine, specify a positive value.
- To tighten the timing constraints, specify a negative value.

This parameter is honored by timing optimization (`optDesign`) and timing analysis (`report_timing`). Using this parameter might have an impact on run time during timing optimization.

Tip

Use this parameter together with the `-effort` parameter to direct timing optimization focus.

`-slackAdjustmentPriority` *`integer`*

Provides priority to a path group for the slack adjustment feature. The paths that are a part of overlapping path groups get slack adjustment from the path group that has a higher priority value. If the number is higher, the priority is higher. In case of multiple overlapping path groups with different slack adjustment values, the most constraining slack adjustment is applied.

Valid values of this option range from 0 to 10.

Default: `0`

`-targetSlack` *`float`*

Specifies a target value for setup slack, in nanoseconds, for all paths that are part of a specific path group. The value can be positive or negative.

This parameter is honored by timing optimization (`optDesign`). Using this parameter might have an impact on run time during timing optimization.

The path group target slack and the global setup target slack (`setOptMode -setupTargetSlack`) are cumulative:

- The path group target slack applies a slack adjustment with the reverse value.
- The global setup target slack is the timing closure target for timing optimization.

Examples

- To add 2 nanoseconds of slack adjustment to all paths belonging to the `PATH_GROUP_A` path group, specify the following command:

  ```
  setPathGroupOptions PATH_GROUP_A -slackAdjustment 2
  ```

- To subtract 200 picoseconds from all paths belonging to `PATH_GROUP_B` path group, specify the following command:

  ```
  setPathGroupOptions PATH_GROUP_B -slackAdjustment -0.2
  ```

- To ensure that the `optDesign` command does not try to achieve a better slack than 2 nanoseconds on path group `PATH_GROUP_A`, specify the following commands:

  ```
  setPathGroupOptions PATH_GROUP_A -targetSlack 2
  optDesign
  ```

When these commands run, the software internally subtracts 2 nanoseconds of slack adjustment from all paths that belong to PATH_GROUP_A path group.

setSchedulingFile

`setSchedulingFile` *`schedulingfileName`*

Specifies the scheduling file. The file is not taken into account by CTS, but `skewClock` must have access to the file if it runs several times consecutively.

To see the current setting for the `setSchedulingFile` command, use `getSchedulingFile`.

Parameters

`schedulingFileName`

Specifies the name of the scheduling file.

Command Order

Run this command before you run `specifyClockTree`.

Example

The following command sets the name of the scheduling file to `scheduling_file.cts`:

```
setSchedulingFile scheduling_file.cts
```

For more information about how the scheduling file is used in the pre-CTS flow for `skewClock`, see skewClock on page 2120.

setUsefulSkewMode

```
setUsefulSkewMode
     [-help]
     [-reset]
     [-allNegEndPoints {true | false}]
     [-ecoRoute {true | false}]
     [-maxAllowedDelay delay]
     [-maxSkew {true | false}]
     [-noBoundary {true | false}]
     [-useCells {cellNames …} ]
```

Sets global parameters for the skewClock command.

Use the getUsefulSkewMode to return the current settings for the setUsefulSkewMode command.

Parameters

-allNegEndPoints {true |false}

Skews all negative path flip-flops, regardless of the minimum achievable target slack.
Default: false

Important

Use this parameter only if you are an expert user, and only on designs with small negative slack (range of -300 ps maximum).

-ecoRoute {true | false }

Uses the NanoRoute detailed router to route nets that are added or changed during useful skew optimization. You must specify setOptMode -effort high to enable this parameter.
Default: false

-help

Outputs a brief description that includes type and default information for each setUsefulSkewMode parameter.

For a detailed description of the command and all of its parameters, use the man command:
man setUsefulSkewMode.

`-maxAllowedDelay` *`delay`*

Limits the amount of slack (in nanoseconds) that the software can borrow from neighboring flip-flops when performing useful skew operations. The Encounter delay calculation and RC extraction methods might differ from those of sign-off tools, so other setup violations might occur if the software borrows too much slack. By having control over slack borrowing, you can prevent these setup violations. Limiting borrowed skew also limits the clock tree skew to avoid large hold violations. The `-maxAllowedDelay` and `-maxSkew` parameters are complementary: If you specify both parameters, the software uses the aggressive algorithm while respecting the `-maxAllowedDelay` limit.
Default: `2.14748e+09`
Value Range: `0.000000 – 2147483647.000000`

`-maxSkew {true | false}`

Advances sequential elements more aggressively than it does by default, without degrading the worst negative slack. The `-maxAllowedDelay` and `-maxSkew` parameters are complementary: If you specify both parameters, the software uses the aggressive algorithm while respecting the `-maxAllowedDelay` limit.
Default: `false`

`-noBoundary {true |false}`

Excludes or includes boundary sequential cells in addition to ordinary sequential elements in useful skew calculations.
Default: `false`

`-reset`

Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setUsefulSkewMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

`-useCells {`*`cellNames …`*`}`

Specifies the cells for `skewClock` to use during post-CTS buffer insertion. Enclose the cell names in braces.
Default: `""` (empty string)

Examples

- The following commands run useful skew using cells `CLKBUFX8, CLKBUFX12, CLKINVX8, CLKINVX12`.

```
setUsefulSkewMode -useCells {CLKBUFX8 CLKBUFX12 CLKINVX8 CLKINVX12}
setOptMode -usefulSkew true
optDesign
```

- The following commands run useful skew with maximum slack borrowing:

```
setUsefulSkewMode -maxSkew true
setOptMode -usefulSkew true
optDesign
```

- The following command resets the `-ecoRoute` parameter to its default value:

```
setUsefulSkewMode -reset -ecoRoute
```

- The following command resets all `setUsefulSkewMode` parameters to their default values:

```
setUsefulSkewMode -reset
```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Useful Skew"

skewClock

```
skewClock
     [-postRoute]
```

Modifies the clock arrival time on sequential elements in order to improve the data path timing between two sequential elements. `skewClock` outputs depend on whether the clock tree is built.

- In pre-CTS mode, `skewClock` generates two files: a scheduling file and latency file.
- In post-CTS mode, `skewClock` inserts buffers and/or inverters in an existing clock tree.

Important

Cadence recommends using `setOptMode -usefulSkew` and `optDesign -preCTS` or `-postCTS` to run useful skew optimization during timing optimization. For information on how to use `optDesign`, see "Optimizing Timing" in the *Encounter User Guide*.

Pre-CTS mode

The `skewClock` command identifies elements that can be advanced or delayed, and computes target delay for each. An advancable element is a sequential gate on which the clock signal can be safely made to arrive early without worsening the slack on the critical path. In general, this is true for sequential elements where the slack on the output pin (SQ) is negative (SQ < 0) and is worse than the slack on the input pin (SD) : (SQ < SD).

The `skewClock` command generates the following files:

- Scheduling file

  The CTS scheduling file lists the delays to consider on sequential elements by the clock tree synthesis tool. When loaded in CTS, the scheduling file constrains CTS to build a tree that fulfills the targets computed by `skewClock`.

  The scheduling file must be loaded in addition to the original clock tree specification file (see "Examples" section below).

  The syntax of the scheduling file is as follows:

  ```
  Macromodel pin lefPinName maxRiseDelay minRiseDelay maxFallDelay
  minFallDelay inputCap
  ```

  An entry in the configuration file is available to specify the scheduling file:

  ```
  set rda_input ui_scheduling_file scheduling_file.cts
  ```

When the design is saved, the scheduling file is copied in the design directory, and the line above is added to the configuration file of the saved design.

- Latency file

  This file lists the delays to considered on sequential elements by the timing analysis tool. These delays are computed by `skewClock`.

  The file syntax is as follows:

  `set_clock_latency rise` *`delay pinName`*

  The timing engine takes the latency file into account when the following parameter is specified:

  `setAnalysisMode -usefulSkew true`

  When building the timing graph, the Encounter software first loads the timing constraints, and then the latency file, as specified in the configuration file, for example

  `Set_rda_input (ui_latency_file) latency_file.sdc`

  When the design is saved, the latency file is copied to the design directory, and the line above is added to the configuration file of the saved design.

Post-CTS Mode

The `skewClock` command identifies delayable elements and delays the clock for each element. A delayable element is a sequential element on which the clock signal can be safely delayed without worsening the slack on the critical path. In general, this is true for each sequential element where the slack on the data pin (SD) is negative (SD<0) and is worse than the slack on the output pin (SQ) : (SD < SQ).

The `skewClock` command inserts buffers and/or inverters on an existing clock tree.

The following commands are related to `skewClock`:

- `getLatencyFile`
- `getSchedulingFile`
- `getUsefulSkewMode`
- `setLatencyFile`
- `setNanoRouteMode`
- `setSchedulingFile`
- `setUsefulSkewM`

Parameters

`-postRoute`

Forces the command to end with ECO routing for the modified clock nets. This parameter is mandatory in order to use the `skewClock` command in the postroute stage.
Default: `false`

Examples

- The following commands show how to use `skewClock` in a pre-CTS flow without using `optDesign`:

```
setAnalysisMode -usefulSkew true
reportViolation num 1
cleanupSpecifyClockTree
setSchedulingFile scheduling_file.cts
skewClock
reportViolation num 1
specifyClockTree -clkfile myCTS_constraintfile.cts
specifyClockTree -clkFile scheduling_file.cts
ckSynthesis
reportViolation num 1
```

  Note: The recommended method for useful skew in pre-CTS mode is as follows:

```
setOptMode -usefulSkew true
optDesign -preCTS
```

Related Topics

- Optimizing Timing chapter in the *Encounter User Guide*
  - "Using Useful Skew"

37

Timing Modeling Commands

compare_model_timing

```
compare_model_timing
     -ref reference_filename
     -compare filename
     -outFile output_filename
     [-percent_tolerance tolerance_value]
     [-percent_cap_tolerance value]
     [-absolute_cap_tolerance value]
     [-percent_tran_tolerance value]
     [-absolute_tran_tolerance value]
     [-absolute_tolerance tolerance_value]
```

Compares two reports that contain interface timing characteristics generated by the write_model_timing command.

This command compares the interface timing report generated by using the write_model_timing command on the original netlist with the timing report generated by using the write_model_timing command on the model extracted by the do_extract_model command for the same design.

Parameters

`-absolute_cap_tolerance value`

Specifies the tolerance value for capacitance (as an absolute value) while comparing two report files. This parameter specifies the permissible difference, as an absolute value, between the capacitance values of two report files. If the difference in the value of capacitance exceeds the absolute tolerance value specified, the comparison status is set to `FAIL`.

Note: When this parameter is not specified, the software uses the absolute value specified with the `-absolute_tolerance` parameter. If the `-absolute_tolerance` parameter is not specified, the value defaults to `0`.

Important

When the `-percent_cap_tolerance` and `-absolute_cap_tolerance` parameters are used together, the comparison status is set to `FAIL` only when the difference in the value of capacitance exceeds the specified absolute as well as percentage tolerance values.

`-absolute_tolerance` *`tolerance_value`*

Specifies the absolute tolerance value. The tolerance value is used when there is a difference between the timing characteristics of the reference file (`-ref`) and the file that is being compared (`-compare`) to determine the PASS or `FAIL` status of the comparison. If the difference in the value of a timing characteristic exceeds the absolute tolerance value, the comparison status is set to `FAIL`.

Note: When used with the `-percent_tolerance` option, the difference in the value of a timing characteristic should exceed both absolute and percentage tolerance values for the comparison result status of an arc to be `FAIL`.

Default: `0`

`-absolute_tran_tolerance` *`value`*

Specifies the tolerance value for transition time at ports (as an absolute value) while comparing two report files. This parameter specifies the permissible difference, as an absolute value, between the transition time at ports of two report files. If the difference in the transition time exceeds the absolute tolerance value specified, the comparison status is set to `FAIL`.

Note: When this parameter is not specified, the software uses the absolute value specified with the `-absolute_tolerance` parameter. If the `-absolute_tolerance` parameter is not specified, the value defaults to `0`.

Important

When the `-percent_tran_tolerance` and `-absolute_tran_tolerance` parameters are used together, the comparison status is set to `FAIL` only when the difference in the value of transition exceeds the specified absolute as well as percentage tolerance values.

`-compare` *`filename`*

Specifies the name of the report file that contains the interface timing characteristics extracted from the model created using the `do_extract_model` command. The timing characteristics of this file are compared with the timing characteristics of the reference file that you specify with the `-ref` option.

`-outFile` *`output_filename`*

Specifies the name of the output file in which the comparison results will be generated. This file contains the PASS or `FAIL` status for the timing data comparison.

`-percent_cap_tolerance` *`value`*

Specifies the tolerance value for capacitance (as a percentage) while comparing two report files. This parameter specifies the permissible difference, in percentage, between the capacitance values of two report files. If the percentage difference in the value of capacitance exceeds the percentage tolerance value specified, the comparison status is set to `FAIL`.

Note: When this parameter is not specified, the software uses the percentage value specified with the `-percent_tolerance` parameter. If the `-percent_tolerance` parameter is not specified, the value defaults to `0`.

Important

When the `-percent_cap_tolerance` and `-absolute_cap_tolerance` parameters are used together, the comparison status is set to `FAIL` only when the difference in the value of capacitance exceeds the specified absolute as well as percentage tolerance values.

`-percent_tolerance` *`tolerance_value`*

Specifies the percentage tolerance value for comparing the timing data of two report files. The percentage tolerance value is used when there is a difference between the timing characteristics of the reference file (`-ref`) and the file that is being compared (`-compare`) to determine the PASS or `FAIL` status of the comparison. If the percentage difference in the value of a timing characteristic exceeds the percentage tolerance value, the comparison status is set to `FAIL`.

Note: When used with the `-absolute_tolerance` option, the difference in the value of a timing characteristic should exceed both percentage and absolute tolerance values for the comparison result status of an arc to be `FAIL`.

Default: `0`

`-percent_tran_tolerance` *`value`*

Specifies the tolerance value for transition time at ports (as a percentage) while comparing two report files.This parameter specifies the permissible difference, in percentage, between the transition time at ports of two report files. If the percentage difference in the transition time exceeds the percentage tolerance value specified, the comparison status is set to `FAIL`.

Note: When this parameter is not specified, the software uses the percentage value specified with the `-percent_tolerance` parameter. If the `-percent_tolerance` parameter is not specified, the value defaults to `0`.

Important

When the `-percent_tran_tolerance` and `-absolute_tran_tolerance` parameters are used together, the comparison status is set to `FAIL` only when the difference in the value of transition exceeds the specified absolute as well as percentage tolerance values.

`-ref` *`reference_filename`*

Specifies the name of the reference file that is used for comparison. This is the report file that was generated by using the `write_timing_model` command on the original design.

The file specified with the `-compare` option is compared with this file.

Example

- The following command compares the timing characteristics of the `sample_compare.rpt` file with the `sample_reference.rpt` file and generates the comparison results in the `sample_output.rpt` file:

```
compare_model_timing -ref sample_reference.rpt -compare sample_compare.rpt
-outFile sample_output.rpt
################################################################################
# compare_model_timing report
# Design : netlist =  topcell =
# Version : 07.10-dev.indfe_cm
# Analysis Mode : setup
# Date : Wed Mar 14 15:56:59 IST 2007
################################################################################
```

```
#################################################################################
#Section 1: Worst-case Arc-Delay
#Tolerance Percent: 0.0%
#Tolerance Absolute: 0.0
#NOTE: The various columns list values in the form 'X[Y]', where:
#      X = value that is compared.
#      Y = reference value.

#NOTE2: If a comparison fails (ie. status = FAIL), then the clauses
#       (ie. 'X[Y]' pairs) that caused the failure are identified with
#       a '*' in front of them.
#From   To    Arc-Type   Transition   Arc-Delay   abdDiff   %Diff  Status
#################################################################################
in      CLK3   setup      fall/rise   0.134[0.134]   0.000   0.000%  PASS
in     CLK3   setup       rise/rise    -0.005[-0.005]   0.000   -0.000%  PASS

#################################################################################
#Section 2: Pin-capacitances
#Tolerance: 0.0%
#NOTE: The various columns list values in the form 'X[Y]', where:
#      X = value that is compared.
#      Y = reference value.

#NOTE2: If a comparison fails (ie. status = FAIL), then the clauses
#       (ie. 'X[Y]' pairs) that caused the failure are identified with
#       a '*' in front of them.
#
#Pin              C(total)[rise]         C(total)[fall]           Status
#################################################################################
clk             0.002[0.002]    0.002[0.002]    PASS
in              0.002[0.002]    0.002[0.002]    PASS
out             0.000[0.000]    0.000[0.000]    PASS

#################################################################################
#Section 3: Transition-times
#Tolerance: 0.0%
#NOTE: The various columns list values in the form 'X[Y]', where:
#      X = value that is compared.
#      Y = reference value.

#
```

```
#NOTE2: If a comparison fails (ie. status = FAIL), then the clauses
#       (ie. 'X[Y]' pairs) that caused the failure are identified with
#       a '*' in front of them.
#
#Pin            Rise            Fall            Status
#############################################################################
clk             0.000[0.000]    0.000[0.000]    PASS
in              0.000[0.000]    0.000[0.000]    PASS
out             0.056[0.056]    0.051[0.051]    PASS

#############################################################################
#Section 4: Design Rules
#Tolerance: 0.0%
#NOTE: The various columns list values in the form 'X[Y]', where:
#      X = value that is compared.
#      Y = reference value.

#NOTE2: If a comparison fails (ie. status = FAIL), then the clauses
#       (ie. 'X[Y]' pairs) that caused the failure are identified with
#       a '*' in front of them.
#
#Pin    MaxCap    MinCap    MaxTrans    MaxFanout    FanoutLoad    Status
#############################################################################
clk     NA[NA]     NA[NA]    NA[NA]      NA[NA]       NA[NA]        PASS
in      NA[NA]     NA[NA]    NA[NA]      NA[NA]       NA[NA]        PASS
out     NA[NA]     NA[NA]    NA[NA]      NA[NA]       NA[NA]        PASS

#############################################################################
# SUMMARY
#
# Check-Type    Passed      Failed             N/A            Total
#############################################################################
Worst-Slacks    2               0               0               2
Capacitances    3               0               0               3
Trans-Times     3               0               0               3
Design-Rules    3               0               0               3
```

create_ilm_data_dir

```
createILMDataDir
     -dir directory_name
     [-cell cell_name]
     [-setup | -hold | -setupHold | -mmmc]
     [-verilog cell_name.v]
     [-spef spef_file.spef]
     [-sdc sdc_file.sdc]
     [-viewName view_name]
     [-rcCornerName RC_corner_name]
     [-overwrite]
     [-incr]
     [-cts]
     [-si]
```

Creates an ILM directory from existing data.

If the data files already exist in the specified directory, the tool stops and generates an error message. You can overwrite existing files by specifying `-overwrite`. You can add files to the directory by specifying `-incr`.

You can specify that the software uses the data in the directory for setup, hold, MMMC, or both setup and hold analysis.

Parameters

`-cell` *`cell_name`*	Specifies the name of the ILM module. If you do not specify a cell name, the software uses the `-verilog` *`cell_name`* for the module name.
`-cts`	Creates and stores data in the directory for clock tree synthesis (CTS).
`-dir` *`dir_name`*	Stores the ILM data in the specified directory: the absolute directory path or the path relative to the current directory.
`-hold`	Specifies that the data in the directory will be used for hold analysis. You cannot specify `-setupHold`, `-setup`, `-mmmc`, `-viewName`, or `-rcCornerName` when you use this parameter.
`-incr`	Lets you add files in addition to the existing files to the directory.
`-mmmc`	Specifies that the data will be used for MMMC analysis.

`-overwrite` Overwrites existing files in the directory.

`-rcCornerName` *`RC_corner_name`*

Specifies the RC corner name for MMMC analysis. You cannot specify `-setup`, `-hold`, or `-setupHold` when you use this parameter. The `-spef` parameter identifies the spef file to be used with the specified corner.

`-sdc` *`sdc_name.`*`sdc`

Specifies the name of the `.sdc` file for the analysis view.

`-setup` Specifies that the data in the directory will be used for setup analysis. You cannot specify `-hold`, `-setupHold`, `-mmmc`, `-viewName`, or `-rcCornerName` when you use this command.

`-setupHold` Specifies that the data in the directory will be used in both setup and hold analysis. You cannot specify `-hold`, `-setup`, `-mmmc`, `-viewName`, or `-rcCornerName` when you use this command.

`-si` Specifies that the data in directory will be used for signal integrity analysis.

`-spef` *`spef_name`*`.spef`

Specifies an existing `.spef` file that you want to include in the directory.

`-viewName` *`view_name`*

Specifies the MMMC view for the ILM module. You cannot specify `-setup`, `-hold`, or `-setupHold`. Specify the `-sdc` file for the analysis view.

`-verilog` *`netlist_name`*`.v`

Specifies the verilog file you want to include in the ILM data directory.

Examples

- The following commands create two module directories:

```
create_ilm_data_dir -dir A -verilog V1.v -spef speffile1.spef -sdc sdcfile1
create_ilm_data_dir -dir A -verilog V2.v -spef speffile2.spef -sdc sdcfile2
```

  The command creates two tcl files in the current directory.

do_extract_model

```
do_extract_model
     [-lib_name lib_name]
     [-cell_name cell_name]
     [-clock_slews {clk_slew1 clk_slew2 ...}]
     [-input_slews {input_slew1 input_slew2 ...}]
     [-output_loads {output_load1 output_load2 ...}]
     [-precision value]
     [-gain integer]
     [-resolution value]
     [-tolerance value]
     [-keep_trigger_arcs]
     [-blackbox]
     [-blackbox_2d]
     [-greybox]
     [-assertions constraint_filename]
     [-verilog_shell_file filename]
     [-verilog_shell_module top_module_name]
     [-max_num_slews value]
     [-max_num_loads value]
     [-traverse_borrowing_latch]
     model_filename
```

Builds a Liberty (`.lib`) format model for the top cell, which is the timing model equivalent of the original design. The model is extracted for the given analysis modes, PVT conditions, and parasitics. If any one of these parameters change, then you must extract the model again. The model is extracted for a range of input slews and load capacitances. If you do not specify these ranges, the software automatically computes the default values. The extracted model defines all of the top-level ports of the design as pins. Additionally, all other design pins that are targets of `create_clock` or `create_generated_clock` are retained as internal pins.

Tip

For model extraction, your design must be properly constrained. For example, no clock or data conflicts can be present. Use the `check_timing` command and correct all errors prior to model extraction.

Parameters

`-assertions` *`constraint_filename`*

Dumps the false and multicycle path exceptions associated with the internal pins, which will be preserved in the grey box models. The exceptions are dumped in SDC format and can be used while instantiating grey box models.

Default: `model.asrt`

`-blackbox`

Generates a black-box extracted model, which can be used by tools that do not support internal pins. Black-box models are primarily used for IP characterization. These models do not support false and multicycle path exceptions. Therefore, this option should be used only when the design does not contain false and multicycle path exceptions.

`-blackbox_2d`

Transforms three dimensional tables to two dimensional tables by correcting the capacitance at the secondary output to the current value.

`-cell_name` *`cell_name`*

Specifies the cell name used in the timing model.

Default: The name of the top cell to be extracted.

`-clock_slews {`*`clk_slew1 clk_slew2...`*`}`

Specifies a range of slew values used at all clock pins. The list of values must be enclosed by curly braces. If this parameter is not specified, the software automatically computes the appropriate values.

`-gain` *`integer`*

Represents an increase in the number of delay arcs if a pin is removed. This option reduces the extracted model size in scenarios where removing a pin can result in increased model size. The gain at an internal pin can be defined as:

```
(number of incoming delay arcs * number of outgoing delay arcs) - (number of incoming delay arcs + number of outgoing delay arcs
```

If an internal pin has a gain greater than the specified number, then that pin needs to be preserved in order to reduce the extracted model size.

Note: This option works only when the `-greybox` option is used.

Default: `0`

`-greybox`

Extracts a grey box model for the design instead of a black-box model. A grey box model can be defined as a model that contains internal pins.

The software provides support for path exceptions in extracted timing models by retaining some internal pins associated with these exceptions. You can use this option to extract a design with path exceptions on its interface paths.

`-input_slews {`*`input_slew1 input_slew2 ...`*`}`

Specifies a range of slew values used at all input pins. Enclose the list of values with curly braces. If this parameter is not specified, the software automatically computes the appropriate values.

`-keep_trigger_arcs` Models clock and data signal conflicts by preserving trigger arcs. A trigger arc is an arc from a clock pin to a register or latch output pin, which arises when a data signal arrives at a pin instead of a clock signal.

This option is used for characterizing designs that are not completely constrained, which means designs in which the clock definitions are either incomplete or missing.

The `-keep_trigger_arcs` option allows you to define clocks after model extraction and helps yield timing results as if the clocks were defined prior to performing model extraction.

When this option is not used, the model extractor expects the clock definitions to be complete and removes the relevant trigger arcs and timing checks if any clock and data signal conflicts occur. This prevents data signals from propagating to the clock paths.

Note: This option works only when the `-greybox` option is used and is not supported with the `-blackbox` option.

`-lib_name` *`lib_name`*

Specifies the library name used in the timing model. Write extracted timing models in the liberty format by specifying the `.lib` file suffix.

Default: The name of the top cell to be extracted.

`-max_num_loads` *`value`*

Specifies the maximum number of output load values used for delay arc characterization. A low value of 1 or 2 improves runtime and memory usage but may significantly degrade model accuracy. Use this parameter only if the model size or model extraction time is critical.

Default: 64

`-max_num_slews` *`value`*

Specifies the maximum number of input slews or clock slews used for delay and check arc characterization. Specifying a low value of 1 or 2 improves runtime and memory but may significantly degrade model accuracy. Using the `-max_num_slews` parameter has a larger impact on runtime and memory than the `-max_num_loads` parameter. Use this parameter only if the model size or the model extraction time is critical.

Default: `64`

The `max_num_slews` and `max_num_loads` parameters specify only the maximum limits on how many slew and load points are chosen. The software scans the gates connected to the input and output ports to calculate the number of points.

`model_filename` Specifies the name of the output file.

Avoid problems with rebind by using the `-lib_name` and `-cell_name` parameters instead of using the default names.

`-output_loads {`*`output_load1 output_load2 ...`*`}`

Specifies a range of load values used at all output pins. Enclose the list of values by curly braces. If this parameter is not specified, the software automatically computes the appropriate values.

`-precision` *`value`* Specifies the floating point precision used for the timing model.

For example, the floating point precision for 3.0089 can be changed as follows:

Precision	Data Value in .lib
4	3.0089
3	3.009
2	3.01
1	3.0

Default: `4`

`-resolution` *`value`*
: Specifies the magnitude of delay difference that can be ignored by the extractor.

  Default: `0.001`

  Type: Float

`-tolerance` *`value`*
: Specifies the accuracy level of extractor in percent. For example, 5.0 represents a plus or a minus 5.0 percent allowable error in the extracted model. The higher the tolerance, the faster the extraction time.

  Default: `1.0`

  Type: Float

`-traverse_borrowing_latch`
: Enables the latch-based extraction of timing models. This functionality is particularly useful for designs where transparent latches are present on the interface. Transparent latches allow propagation from D-pin to Q-pin only if the arrival time on the D-pin is later than the rising or falling edge of the clock for the latch. When borrowing occurs, this parameter helps determine the timing beyond the transparent latch through the D-Q arc. When you specify this parameter, model extraction does not use the actual time borrowed at the latch. Instead, the latch is traced through the D-Q arc to the next level.

`-verilog_shell_file` *`filename`*
: Generates a Verilog shell file that instantiates the timing model. This parameter is useful for validating the model. Use the `-verilog_shell_module` parameter to specify the name of the top module in this Verilog file.

`-verilog_shell_module` *`top_module_name`*
: Specifies the top module name of the Verilog shell file.

  Default: `test_shell`

Examples:

- The following command builds a `.lib` model (`block.lib`) for the block:

  ```
  do_extract_model block.lib
  ```

- The following command creates the timing model for the `test_model` cell belonging to the `test_lib` library, and writes the model into the `test.lib` liberty file:

  ```
  do_extract_model -cell_name test_model -library_name test_lib test.lib
  ```

- The following commands generate timing models in setup and hold mode:

  ```
  set_analysis_mode -setup
  do_extract_model block_setup.lib
  set_analysis_mode -hold
  do_extract_model block_hold.lib
  ```

write_model_timing

```
write_model_timing
     -type { arc | slack }
     -slew_propagation { worst | path_based }
     [-verbose]
     [-nworst value ]
     [-inputs {port_list}]
     [-outputs {port_list}]
     [-traverse_borrowing_latch]
     model_timing_report_file
```

Writes the interface timing characteristics of the design in the specified timing model report file.

The report file contains the following sections:

- `Worst-Case Arc/Slack`: Contains details about the extracted arcs and the slack or delay across them depending on the argument specified with the `-type` option.
- `Transition Time`: Contains information about transition time at ports.
- `Capacitance`: Contains the capacitance values for the ports.
- `Design Rules`: Contains the design rules applied to the ports.

Parameters

`model_timing_report_file`

Specifies the name of the interface timing model report file that will be generated with the interface timing characteristics of the loaded design.

`-inputs {`*`port_list`*`}`

Restricts the `write_model_timing` report to the input ports specified with this parameter. As a result, the software considers the timing paths related to the specified input ports only, while generating the report.

If this parameter is not specified, all the input ports in the design are considered while generating the report.

`-nworst` *`value`*
: Specifies the number of paths to be enumerated for each end point.

  Note: The extracted model will contain greater number of setup/hold paths per end point because it will have lesser end points compared to the original netlist. Therefore, it is recommended that you use a greater `-nworst` value such as `40` while writing a timing report for the extracted model to derive better correlation.

`-outputs {`*`port_list`*`}`
: Restricts the `write_model_timing` report to the output ports specified with this parameter. As a result, the software considers the timing paths related to the specified output ports only, while generating the report.

  If this parameter is not specified, all the output ports in the design are considered while generating the report.

`-slew_propagation { worst | path_based }`
: Sets the slew propagation mode to use for dumping the timing characteristics of the design.

`-traverse_borrowing_latch`
: Enables the latch-based validation of timing models.

  Use this parameter on the original netlist before comparing the generated report with the model extracted using the `-traverse_borrowing_latch` parameter of the `do_extract_model` command.

`-type { arc | slack }`
: Determines whether to report arc delay values (`arc`) or the worst-slack values (`slack`).

`-verbose`
: Prints the command progress details.

Example

- The following command extracts the interface timing characteristics of the currently loaded design and writes out the report file named `sample.rpt` with the arc delay information:

```
write_model_timing -type arc sample.rpt
##############################################################################
```

```
# write_model_timing report
# Design : netlist = .//case1.v topcell = set_propagated_clock_tc2
# Version : 07.10-dev.indfe_cm
# Analysis Mode : setup
# Date : Wed Mar 14 15:29:57 IST 2007
#############################################################################
#############################################################################
# Section 1 : Worst Case Arc
# Analysis Mode : setup
# Description : worst case arcs
#
# From         To             Arc Type           Transition        Arc-Delays
#############################################################################
in          CLK3            setup           fall/rise       0.134
in          CLK3            setup           rise/rise       -0.005
#####################################################
# Section 2 : Transition time
# Analysis Mode : setup
# Description : Actual transition times on the ports
#
# Port     RiseTran    FallTran
#####################################################
in              0.000           0.000
clk             0.000           0.000
out             0.056           0.051
#####################################################
# Section 3 : Capacitance
# Analysis Mode : setup
# Description : Total Cap on the ports
#
# Port     CTotal(Rise)        CTotal(Fall)
#####################################################
in              0.002           0.002
clk             0.002           0.002
out             0.000           0.000
#############################################################################
# Section 4 : Design Rules
# Analysis Mode : setup
# Description : Design Rules on the ports
#
# Port      MaxCap      MinCap       MaxTrans      MaxFanout       Fanout Limit
```

```
#############################################################################
in              NA              NA              NA              NA              NA
clk             NA              NA              NA              NA              NA
out             NA              NA              NA              NA              NA
```

38

Advanced Timing Tcl Scripting Commands

add_to_collection

```
add_to_collection
     base_collection
     second_collection_or_list
     [-unique]
```

Creates a new collection by adding objects from one collection to another collection. The following rules apply:

- If all of the objects in the base collection are of the same type, only the objects in the second collection that are the same type as those in the base collection are added to the new collection. If you specify an object list instead, all of the objects in the list must be the same type as those in the base collection.
- If the base collection contains different types of objects, all objects in the second collection are added. In this situation, you cannot specify an object list.

If the software finds no objects to add, an empty collection is returned.

Parameters

base_collection	Specifies the base collection of objects. Objects from the second collection or object list are added to this collection to create the new collection.
second_collection_or_list	Specifies the second object collection, or a list of objects.
-unique	Removes any duplicate objects from the new collection.

all_clocks

```
all_clocks
```

Creates a collection of all the clocks in the current design. If no clocks were defined, an empty string `""` is returned.

To filter clocks, use the `get_clocks` command to create a collection of clocks matching a specific pattern. Assign these clocks to a variable or pass them using another command.

Example

- The following example applies the `set_propagated_clock` command to all the clocks in the design:

  ```
  set_propagated_clock [all_clocks]
  ```

Related Information

`create_clock`

`get_clocks`

`set_propagated_clock`

`all_inputs`

all_connected

`all_connected [-leaf] *single_object_collection_or_object*`

Returns a collection of all of the objects connected to the specified net, pin, or port.

Parameters

`-leaf`
: Returns only the flattened connection points (leaf pins) for the specified net.

  Default: Returns all hierarchical ports connected to the specified net

`single_object_collection_or_object`
: Specifies the net, pin, or port for which to return the collection of connected objects. You can specify an object name, or a single-object collection.

Examples

- The following command returns a collection of all of the objects connected to the pin `b1_Sin1`:

  ```
  all_connected Sin1
  ```

  The software returns the following information:

  ```
  Sin1
  ```

- The following command returns a collection of all of the objects connected to a pin contained in the single collection object `i_block1/b1_Sin1`:

  ```
  all_connected [get_pins {i_block1/b1_Sin1}]
  ```

  The software returns the following information:

  ```
  Sin1
  ```

- The following command returns a collection of all of the hierarchical ports connected to the net `net_1`:

  ```
  all_connected [get_nets net_1]
  ```

  The software returns the following information:

  ```
  TL2/l2_in TL1/l1_out
  ```

- The following command returns a collection that contains only the flattened connection points for the net `net_1`:

```
all_connected -leaf [get_nets net_1]
```

The software returns the following information:

```
TL2/L1/RX/D TL1/RX/Q
```

all_fanin

```
all_fanin
     -to {collection | object_list}
     [-levels value]
     [-pin_levels value]
     [-startpoints_only]
     [-only_cells]
     [> file_name]
```

Returns a collection of pins, ports, or cells that exist in the fanin cone of the specified objects.

Parameters

`> file_name`	Stores the generated collection in the specified file. You can add the `.gz` extension to the file name to generate a compressed report. **Note:** The file name must be the last parameter specified.
`-levels value`	Specifies the number of gate levels to go back from a specific pin object to create the collection.
`-only_cells`	Returns a collection only of the cells in the fanin zone.
`-pin_levels value`	Specifies the number delay arcs to go back from a specific pin object to create the collection.
`-startpoints_only`	Returns a collection only of valid start points in the fanout zone.
`-to {collection | object_list}`	Specifies the objects for which to return the collection. You can specify a list of objects, or a collection of objects.

all_fanout

```
all_fanout
     -from {collection | object_list}
     [-levels value]
     [-pin_levels value]
     [-endpoints_only]
     [-only_cells]
     [> file_name]
```

Returns a collection of pins, ports, or cells that exist in the fanout cone of the specified objects.

Parameters

`> file_name`	Stores the generated collection in the specified file. You can add the `.gz` extension to the file name to generate a compressed report. **Note:** The file name must be the last parameter specified.
`-endpoints_only`	Returns a collection only of valid end points in the fanout zone.
`-levels value`	Specifies the number of gate levels to go back from a specific pin object to create the collection.
`-only_cells`	Returns a collection only of the cells in the fanout zone.
`-pin_levels value`	Specifies the number delay arcs to go back from a specific pin object to create the collection.
`-from {collection | object_list}`	Specifies the objects for which to return the collection. You can specify a list of objects, or a collection of objects.

all_inputs

```
all_inputs
     [-clock list_of_clocks]
     [-no_clocks]
     [-edge_triggered]
     [-level_sensitive]
```

Creates a collection of all the input ports in the current design. To filter inputs, use the `get_ports` command to create a collection of ports matching certain criteria. Assign these ports to a variable or pass them using another command. You can limit the contents of the collection to ports with input delay relative to a specific clock.

Parameters

`-clock` *`list_of_clocks`*

Considers only input ports with input delay relative to the specified list of clocks.

`-edge_triggered` Returns a collection of input ports with the `set_input_delay -clock` constraint.

`-level_sensitive` Returns a collection of input ports with the `set_input_delay -level_sensitive` constraint.

`-no_clocks` Returns primary input ports that do not have `create_clock` or `create_generated_clock` constraints.

Example

- The following command returns the names all of the input ports with input delay relative to an `ideal_clk`:

  ```
  all_inputs -clock ideal_clk
  ```

Related Information

`get_ports`

`report_ports`

`set_input_delay`

all_instances

`all_instances` *`object`*

Returns a collection of all instances available in a design, or all instances of a specified library cell existing in the current design.

You cannot use the `all_instances` command on the top level of the design (`all_instances [get_design top]`).

However, you can use the command for hierarchical modules inside the top level. For example, if SUB is a hierarchical module in the top level of the design, you can specify the following command to return all instances of that module: `all_instances [get_design SUB]`.

Parameters

`object`	Specifies the name of a design or a library cell in the current design. You can specify a pattern for the design or library cell name, or a single collection of a design or library cell.

all_outputs

```
all_outputs
     [-clocks list_of_clocks]
     [-edge_triggered]
     [-level_sensitive]
```

Creates a collection of all output ports in the current design. To filter outputs, use the `get_ports` command to create a collection of ports matching certain criteria. Assign these ports to a variable or pass them using another command. You can limit the contents of the collection to ports with output delay relative to a specific clock.

Parameters

`-clocks` *`list_of_clocks`*

Considers only output ports with an output delay relative to a specific list of clocks.

`-edge_triggered` Returns a collection of output ports with the `set_output_delay -clock` constraint.

`-level_sensitive` Returns a collection of output ports with the `set_output_delay -level_sensitive` constraint.

Example

- The following command returns the names all of the output ports with an output delay relative to an `ideal_clk`:

  ```
  all_outputs -clock ideal_clk
  ```

Related Information

`get_ports`

`report_ports`

`set_output_delay`

all_registers

```
all_registers
     [-clock {clock_list}]
     [-rise_clock {clock_list}]
     [-fall_clock {clock_list}]
     [-flops | -edge_triggered]
     [-latches | -level_sensitive]
     [-macros]
     [-master_slave]
     [-data_pins]
     [-clock_pins]
     [-output_pins]
     [-async_pins]
     [-slave_clock_pins]
```

Returns a collection of instances and pins of flip-flops, latches, or sequential macros in the design. A cell is considered sequential if the library cell defines trigger or timing check arcs.

Note: The following options can be used only after building the timing graph:

- `-clock`
- `-rise_clock`
- `-fall_clock`

Parameters

`-async_pins` Returns a collection of async preset or clear pins.

`-clock {clock_list}`

Specifies the name of a clock or a list of clocks. The `all_registers` command creates a collection of registers clocked by the specified *`clock_list`*.

`-clock_pins` Returns a collection of register clock pins. You can use other parameters to further filter this collection.

`-data_pins` Returns a collection of register data input pins. You can use other parameters to further filter this collection.

`-fall_clock {clock_list}`

Specifies the name of a clock or a list of clocks. The `all_registers` command creates a collection of registers opened by the falling edge of the clocks specified in the *`clock_list`*.

`-flops | -edge_triggered`

Returns a collection of edge-triggered sequential instances. Both the `-flops` and `-edge_triggered` options return the same collection. The `-flops` option is provided for backward compatibility.

`-latches | -level_sensitive`

Returns a collection of level-sensitive sequential instances. Both `-latches` and `-level_sensitive` options return the same collection. The `-latches` option is provided for backward compatibility.

`-macros` Returns sequential instances that are neither flip-flops nor latches.

`-master_slave` Returns a collection of master-slave register cells.

`-output_pins` Returns a collection of register output pins.

`-rise_clock {`*`clock_list`*`}`

Specifies the name of a clock or a list of clocks. The `all_registers` command creates a collection of registers opened by the rising edge of the clocks specified in the *`clock_list`*.

`-slave_clock_pins` Returns a collection of slave clock pins of the master-slave registers. You can specify the slave clock pins as `clocked_on_also` in the library.

Example

- The following command reports the10 worst paths going to registers in the design:

```
report_timing -to [all_registers] -max_paths 10
```

Related Information

`get_cells`

`get_pins`

append_to_collection

```
append_to_collection
     [var_name]
     second_collection_or_list
     [-unique]
```

Appends objects from one collection to another collection. The following rules apply:

- If all of the objects in the base collection are of the same type, only the objects in the second collection that are the same type as those in the base collection are appended to the base collection. If you specify an object list instead, all of the objects in the list must be of the same type as those in the base collection.
- If the base collection contains different types of objects, all objects in the second collection are appended to it. In this situation, you cannot specify an object list.

If the software finds no objects to add, an empty collection is returned.

Parameters

var_name	Specifies a pointer to the base collection of objects. Objects in the second collection or object list are appended to this collection. If you specify a pointer that does not refer to an existing base collection, the software creates a new collection that uses the pointer as a reference.
second_collection_or_list	Specifies the second collection of objects, or an object list.
-unique	Retains any duplicate objects that already exist in the base collection, but does not append duplicate objects from the second collection to the base collection.

Examples

- The following command appends the nets in the specified collection to the collection newVar:

  ```
  append_to_collection newVar [get_nets {CLK1}]
  ```

- The following command appends the specified nets to the collection newVar:

  ```
  append_to_collection newVar {Sin1 CLK1 Sout1}
  ```

compare_collections

```
compare_collections collection1 collection2 [-order_dependent]
```

Compares two collections, and returns a value of `0` if all of the objects contained in both collections are the same. If the objects contained in both collections are different, the software returns a value of `1`.

Parameters

collection1	Specifies the first collection.
collection2	Specifies the second collection.
-order_dependent	Specifies that the order of the objects in both collections must be the same for the collections to be considered the same.

copy_collection

`copy_collection` *`base_collection`*

Returns a collection that is an exact copy of the specified base collection.

Parameter

`base_collection`	Specifies the base collection of objects.

filter_collection

`filter_collection` *`base_collection`* {*`filter_expression`*}

Returns a collection of objects that were filtered from the specified collection of objects based on user-specified criteria (*`filter_expression`*). The following rules apply:

- The *`filter_expression`* can be created by combining several object properties using the following relational and logical operators: >, <, ==, !=, <=, >=, &&, ||, =~, !~, AND, OR. You also can use parentheses to create expressions.
- You can use the `defined` and `undefined` existence operators to determine whether an attribute is defined for an object.i
- All design object properties supported through the `get_property` command can be used for filtering the base collection.

Note: You also can perform filtering operations by specifying the `-filter` parameter with the `get_*` commands.

Parameters

`base_collection`	Specifies the collection of objects from which to filter objects for the new collection.
`filter_expression`	Specifies the criteria with which to filter objects. The software adds any objects in the base collection that match *`filter_expression`* to the new collection.

Examples

- The following command returns a single-object collection containing the pin with the hierarchical name `lat_3/Q`:

```
filter_collection [get_pins] {hierarchical_name == lat_3/Q}
```

- The following command returns a collection of pins that have a maximum slew that is greater than 0.1:

```
filter_collection [get_pins] {slew_max_rise > 0.1}
```

- The following commands create a new collection called `newPaths` that contains all of the paths from the collection `paths` that have a slack value that is greater than -0.806 and less than 0.0:

```
set paths [report_timing -collection -max_paths 10]
set newPaths [filter_collection $paths {(slack > -0.806 && slack < 0.0)}]
```

foreach_in_collection

`foreach_in_collection` *`var_name collection body`*

Iterates through all objects in the specified collection and executes the commands in the specified script.

Parameters

`body`	Specifies the script of commands to execute during each iteration.
`collection`	Specifies the collection of objects through which to iterate.
`var_name`	Specifies the iterator variable. During each iteration, *`var_name`* is set to a collection of exactly one object. Any command that accepts collections can accept *`var_name`* as a value.

Examples

- A script with the following commands would print out the hierarchical names of all of the cells in the design:

```
foreach_in_collection iCell [get_cells *] {
    Puts “Cell = [get_property $iCell hierarchical_name]”
}

Cell = ck_0
Cell = ck_1
Cell = ck_2
Cell = ck_4
```

- The following example shows a more complex use of the `foreach_in_collection` command.

```
set cellCollection [get_cells temp1_reg]
set libCellCollection [get_lib_cells $cellCollection]
set libArc [get_lib_arcs $libCellCollection]

foreach_in_collection arc $libArc {
    set fpin [get_property $libArc from_lib_pin]
    set tpin [get_property $libArc to_lib_pin]
    set sense [get_property $libArc timing_type]
```

```
    set from_pin_name [get_property $fpin name]
    set to_pin_name [get_property $tpin name]
    Puts [format “%25s -> %25s %-20s” $from_pin_name $to_pin_name $sense]
}

    CK -> D           setup_rising
    CK -> D           hold_rising
    CK -> CK          minimum_period
    CK -> CK          minimum_period
    CK -> CK          min_pulse_width

           foreach_in_collection.txt
    CK -> CK          min_pulse_width
    RN -> RN          min_pulse_width
    CK -> RN          recovery_rising
    CK -> RN          removal_rising
    CK -> Q           fallin_edge
    RN -> Q           clear
```

get_arcs

```
get_arcs
     { [-to to_list] [-from from_list]
      | -of_objects object_list}
     [-quiet]
```

Creates a collection of timing arcs. Assign these arcs to a variable, or pass them to another command. If no objects match the criteria, an empty collection is returned.

Note: The timing graph must be built before you can use this command.

Parameters

`-from` *`from_list`*
: Adds all timing arcs that fan in directly to the specified pins or ports to the collection.

  You can specify a list of pins or a list of ports, but you cannot specify a list that contains both pins and ports. You can specify a pattern for the pin or port name, a collection of pins or ports, or a pattern and a collection.

  Note: You cannot specify this parameter with the `-of_objects` parameter.

`-of_objects` *`object_list`*
: Adds all timing arcs that are associated with the specified cells or nets to the collection. You can specify a list of cells or a list of nets, but you cannot specify a list that contains both cells and nets. You can specify a pattern for the cell or net name, a collection of cells or nets, or a pattern and a collection.

  Note: You cannot specify this parameter with the `-to` or `-from` parameters.

`-quiet`
: Suppresses all error and warning messages generated when the `get_arcs` command is run.

`-to` *`to_list`*

Adds all timing arcs that fan out directly from the specified list of pins or ports to the collection.

You can specify a list of pins or a list of ports, but you cannot specify a list that contains both pins and ports. You can specify a pattern for the pin or port name, a collection of pins or ports, or a pattern and a collection.

Note: You cannot specify this parameter with the `-of_objects` parameter.

Examples

- The following command returns a collection of the timing arcs fan in directly to the pin `dff1/CP`:

  ```
  get_arcs -from dff1/CP
  ```

- The following command creates a collection of the timing arcs that are associated with the nets in the specified collection:

  ```
  get_arcs -of_objects [get_nets {n1}]
  ```

get_cells

```
get_cells
     [-hierarchical]
     [-hsc char]
     [-filter expr]
     [-regexp]
     [-nocase]
     [-quiet]
     {patterns | -of_objects object_list}
```

Creates a collection of instances in the current design whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument in another command.

Parameters

`-filter` *`expr`*	Filters the collection with the specified expression. For any cells that match the pattern, the software evaluates the expression based on the attribute of the cell. You can filter cells using any of the cell properties supported through the `get_property` command.
`-hierarchical`	Specifies that the patterns should be matched hierarchically. If the hierarchical name of an instance at a particular hierarchical level matches the patterns, it is included in the result.
`-hsc` *`char`*	Specifies the hierarchical delimiter for patterns. For example, if the hierarchical delimiter is `@`, the `sub/I1@I2` pattern matches instance `I2` within the hierarchy limited at `sub/I1`.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-filter` in order to use this parameter.
`of_objects` *`object_list`*	Creates a collection of all cells associated with the specified pins or nets. You can specify a list of pins or a list of nets, but you cannot specify a list that contains both pins and nets. You can specify patterns for the pin or net names, a collection of pins or nets, or a combination of patterns and a collection.

patterns	Specifies patterns for instance names. Patterns include the * and ? wildcard characters and collections of instances. The get_cells command returns a collection of instances that match the patterns. If no such instance is found, an empty collection is returned.
-quiet	Suppresses all error and warning messages generated when the get_cells command is run.
-regexp	Treats the specified patterns as a regular expression patterns. **Note:** You must specify -filter in order to use this parameter. *Default:* Treats the specified pattern as a wild card

Example

- The following command searches for pins within the hierarchy limited at B:

```
get_cells -hsc @ B@*
```

- The following command searches for cells with set pattern and filter:

```
get_cells “o*” -filter “@ref_lib_cell_name != FD2 && @is_black_box==false”
```

Related Information

get_libs

get_lib_cells

get_lib_pins

get_pins

get_clocks

```
get_clocks
     [-filter expr]
     [-regexp]
     [-nocase]
     [-quiet]
     patterns
```

Creates a collection of clocks whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument to another command.

Parameters

`-filter` *`expr`*	Filters the collection with the specified expression. For any clocks that match the pattern, the software evaluates the expression based on the attribute of the clock. You can filter clocks using any of the clock properties supported through the `get_property` command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-filter` in order to use this parameter.
`patterns`	Specifies patterns for clock names. Patterns include the * and ? wildcard characters, and collections of instances and clocks.
`-quiet`	Suppresses all error and warning messages generated when the `get_clocks` command is run.
`-regexp`	Treats the specified patterns as a regular expression patterns. **Note:** You must specify `-filter` in order to use this parameter. *Default:* Treats the specified pattern as a wild card

Example

- The following example returns all clocks beginning with the name `CK`:

  ```
  get_clocks CK*
  ```

Related Commands

`all_clocks`

get_designs

```
get_designs
     [-quiet]
     patterns
```

Creates a collection of modules and assigns this collection to a variable or pass it as an argument to an another command. The software uses this information during timing analysis.

Parameters

patterns	Specifies patterns for module names. Patterns include the * and ? wildcard characters and collections of modules. The get_designs command returns a collection of modules that match the patterns. If no such module is found, an empty collection is returned.
-quiet	Suppresses all error and warning messages generated when the get_design command is run.

Example

- The following command returns a complete list of modules:

  ```
  get_designs *
  ```

get_generated_clocks

```
get_generated_clocks
     [-filter expr]
     [-regexp | -exact]
     [-nocase]
     patterns
```

Creates a collection of generated clocks whose name matches the supplied pattern list. Assign this collection to a variable, or pass it as an argument to another command.

Parameters

`-exact`	Performs exact pattern matching against the object name, instead of wild card or regular expression matching. Use this parameter when the object name includes special characters such as `*`, `?`, and `+`. **Note:** If you specify this parameter, you cannot specify `-regexp`.
`-filter` *`expr`*	Filters the collection with the specified expression. For any clocks that match the pattern, the software evaluates the expression based on the attribute of the clock.
`-nocase`	Specifies that pattern matching is not case sensitive.
`patterns`	Specifies patterns for generated clock names. Patterns include the `*` and `?` wildcard characters, and collections of generated clocks.
`-regexp`	Treats the specified pattern as a regular expression. *Default:* Treats the specified pattern as a wild card **Note:** If you specify this parameter, you cannot specify `-exact`.

get_lib_arcs

```
get_lib_arcs
     { [-to to_lib_pins] [-from from_lib_pins]
       | -of_objects {lib_cell_list | timing_arcs} }
     [-filter expr]
     [-quiet]
```

Creates a collection of library timing arcs. Assign these library timing arcs to a variable, or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-filter` *`expr`*
: Filters the collection with the specified expression. For any library arcs that match the pattern, the software evaluates the expression based on the attribute of the library arc

`-from` *`from_lib_pins`*
: Creates a collection of all library arcs that originate from the specified library pins. You can specify a pattern for the pin name, a collection of pins, or a pattern and a collection.

  If you specify this parameter with the `-to` parameter, the software creates a collection of all library arcs that exist between the “to” and “from” library pins.

  Note: You cannot specify this parameter with the `-of_objects` parameter.

`-of_objects {`*`lib_cell_list | timing_arcs`*`}`
: Creates a collection of all library arcs that are associated with the specified library cells or timing arcs.

  You can specify a list of library cells or a list of timing arcs, but you cannot specify a list that contains both library cells and timing arcs. You can specify a pattern for the library cell or timing arc name, a collection of library cells or timing arcs, or a pattern and a collection.

  Note: You cannot specify this parameter with the `-to` or `-from` parameters.

`-quiet`
: Suppresses all error and warning messages generated when the `get_lib_arcs` command is run.

`-to` *`to_lib_pins`*

Creates a collection of all library arcs that terminate at the specified library pins. You can specify a pattern for the pin name, a collection of pins, or a pattern and a collection.

If you specify this parameter with the `-from` parameter, the software creates a collection of all library arcs that exist between the “to” and “from” library pins.

Note: You cannot specify this parameter with the `-of_objects` parameter.

get_lib_cells

```
get_lib_cells
     [-filter expr]
     [-regexp]
     [-nocase]
     [-quiet]
     {pattern_list | -of_objects object_list}
```

Creates a collection of library cells from the loaded libraries whose name matches the supplied pattern list. Assign these library cells to a variable or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-filter` *`expr`*	Filters the collection with the specified expression. For any library cells that match the pattern, the software evaluates the expression based on the attribute of the library cell. You can filter library cells using any of the library cell properties supported through the `get_property` command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-filter` in order to use this parameter.
`-of_objects` *`object_list`*	Creates a collection of all library cells associated with the specified library pins or cells. You can specify a list of pins or a list of cells, but you cannot specify a list that contains both pins and cells. You can specify patterns for the pin or cell names, a collection of pins or cells, or a combination of patterns and a collection.

pattern_list

Returns only those libraries in the collection whose name matches one or more of the patterns specified in the *pattern_list* argument. The characters ? and * are wild cards. The ? character matches any one character while the * character matches zero or more characters. If a library matches multiple patterns, it appears multiple times in the collection.

A pattern can have only one hierarchical separator character. The left part of the pattern is the library name pattern and the right part is the cell name pattern. If there is no separator character, the pattern is compared to cell names in all the libraries.

A pattern can have only one hierarchical separator character, dividing the pattern into one or two parts. If the hierarchical separator is /, the alternatives are:

- *cell_name_pattern*

  Selects all cells whose name matches the *cell_name_pattern* from all libraries.

- *lib_name_pattern/cell_name_pattern*

  Selects all cells whose name matches the *cell_name_pattern* from all libraries whose name matches the *lib_name_pattern*.

-quiet

Suppresses all error and warning messages generated when the get_lib_cells command is run.

-regexp

Treats the specified patterns as a regular expression patterns.

Note: You must specify -filter in order to use this parameter.

Default: Treats the specified pattern as a wild card

Examples

- The following command prints all cells for all libraries:

  ```
  get_lib_cells *
  ```

- The following command prints a collection with all cells in the lca500kv lib library:

  ```
  get_lib_cells lca500kv/*
  ```

- The following command prints a collection with all cells whose names starts with AN in all libraries starting with lca:

```
get_lib_cells lca*/AN*
```

- The following command disables delay arcs on the `EF11L050032A` library cell. The `set_disable_cell_timing` command effects all instantiations of the specified cell, and lets you disable delay arcs from and to internal pins.

  ```
  set_disable_timing  -from WACLK -to RBO1 [get_lib_cells {EF11L050032A}]
  ```

Related Information

`get_cells`

`get_libs`

`get_lib_pins`

get_lib_pins

```
get_lib_pins
     [-filter expr]
     [-regexp]
     [-nocase]
     [-quiet]
     {pattern_list | -of_objects object_list}
```

Creates a collection of library cell pins from the loaded libraries whose name matches the supplied pattern list. Assign these library cell pins to a variable or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-filter` *`expr`*
: Filters the collection with the specified expression. For any library cell pins that match the pattern, the software evaluates the expression based on the attribute of the library cell pin.

  You can filter library pins using any of the library pin properties supported through the `get_property` command.

`-nocase`
: Specifies that pattern matching is not case sensitive.

  Note: You must specify `-filter` in order to use this parameter.

`-of_objects` *`object_list`*
: Creates a collection of all library pins associated with the specified library cells or pins. You can specify a list of cells or a list of pins, but you cannot specify a list that contains both cells and pins. You can specify patterns for the cell or pin names, a collection of cells or pins, or a combination of patterns and a collection.

pattern_list

Returns only those libraries in the collection whose name matches one or more of the patterns specified in the *pattern_list* argument. The characters ? and * are wild cards; The ? character matches any one character while the * character matches zero or more characters. If a library cell matches multiple patterns, it appears multiple times in the collection.

A pattern can have only two hierarchical separator characters, dividing the pattern into one, two, or three parts. If the hierarchical separator is /, the alternatives are:

- *pin_name_pattern*

  Selects all pins whose name matches the *pin_name_pattern* from all cells in the libraries.

- *cell_name_pattern/pin_name_pattern*

  Selects all pins whose name matches the *pin_name_pattern* from all cells whose name matches the *cell_name_pattern* in all libraries.

- *lib_name_pattern/cell_name_pattern/pin_name_pattern*

  Select all pins whose name matches the *pin_name_pattern* from all cells whose name matches the *cell_name_pattern* from all libraries whose name matches the *lib_name_pattern*.

-quiet

Suppresses all error and warning messages generated when the get_lib_pins command is run.

-regexp

Treats the specified patterns as a regular expression patterns.

Note: You must specify -filter in order to use this parameter.

Default: Treats the specified pattern as a wild card

Examples

- The following command returns all pins in all cells for all libraries:

```
get_lib_pins *
```

- The following command returns a collection with all pins for cell AO1 in any library:

```
get_lib_pins AO1/*
```

- The following command returns a collection with all pins whose name starts with `Z` or `Q` for the `AO1` cell in any library:

  ```
  get_lib_pins {AO1/Z* AO1/Q*}
  ```

- The following command returns a collection with all pins for the `AO1` cell in the `lca500kv` lib library:

  ```
  get_lib_pins lca500kv/AO1/*
  ```

- The following command returns a collection with all pins starting with `Z` in all cells whose names starts with `AN` in all libraries starting with `lca`:

  ```
  get_lib_pins lca*/AN*/Z*
  ```

Related Information

`get_libs`

`get_lib_cells`

get_libs

```
get_libs
     [-filter expr]
     [-regexp]
     [-nocase]
     {-of_objects object_list | pattern_list}
```

Creates a collection of loaded libraries whose name matches those specified in the pattern list. Assign these libraries to a variable or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-filter` *`expr`*	Filters the collection with the specified expression. For any libraries that match the pattern, the software evaluates the expression based on the attribute of the library. You can filter libraries using any of the library properties supported through the `get_property` command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-filter` in order to use this parameter.
`-of_objects` *`object_list`*	Creates a collection of libraries that belong to the library cells in a given object list.
`pattern_list`	Returns only those libraries in the collection whose name matches one or more of the patterns specified in the *`pattern_list`* argument. The characters ? and * are wild cards; The ? character matches any one character while the * character matches zero or more characters. If a library matches multiple patterns, it appears multiple times in the collection.
`-regexp`	Treats the specified patterns as a regular expression patterns. **Note:** You must specify `-filter` in order to use this parameter. *Default:* Treats the specified pattern as a wild card

Examples

- The following command displays a list of loaded libraries starting with a or l:

  ```
  get_libs {a* l*}
  ```

- The following command queries all libraries beginning with the letter 1:

  ```
  get_libs 1*
  ```

- The following command creates a collection of libraries that are associated with the library cells in the specified object list:

  ```
  get_libs -of_objects [get_lib_cells *]
  ```

Related Information

`get_lib_cells`

`get_lib_pins`

get_nets

```
get_nets
     [-hierarchical]
     [-hsc char]
     [-filter expr]
     [-regexp]
     [-nocase]
     [-quiet]
     {patterns | -of_objects object_list}
```

Creates a collection of nets in the current design whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument to an another command. If no such net is found, an empty collection is returned.

Parameters

`-filter` *`expr`*	Filters the collection with the specified expression. For any nets that match the pattern, the software evaluates the expression based on the attribute of the net. You can filter nets using any of the net properties supported through the `get_property` command.
`-hierarchical`	Specifies that the patterns should match hierarchically. If the hierarchical name of a net at any hierarchical level matches the patterns, it is included in the result.
`-hsc` *`char`*	Specifies the hierarchical delimiter for patterns.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-filter` in order to use this parameter.
`-of_objects` *`object_list`*	Creates a collection of all nets associated with the specified cells, pins, or ports. You can specify a list of cells, a list of pins, or a list of ports, but you cannot specify a list that contains cells, pins, and ports together. You can specify patterns for the cell, pin, or port names, a collection of cells, pins, or ports, or a combination of patterns and a collection.
`patterns`	Specifies patterns for net names. Patterns include the * and ? wildcard characters and collections of nets.
`-quiet`	Suppresses all error and warning messages generated when the `get_nets` command is run.

`-regexp`	Treats the specified patterns as a regular expression patterns. **Note:** You must specify `-filter` in order to use this parameter. *Default:* Treats the specified pattern as a wild card

Examples

- The following command returns all nets within the hierarchy limited at A:

  ```
  get_nets -hsc @ A@*
  ```

- The following command returns all patterns that match hierarchy and identifies nets to be excluded from optimization.

  ```
  get_nets * -hierarchical -filter “@dont_touch== true”
  ```

Related Information

`get_pins`

get_object_name

`get_object_name` *`single_object_collection`*

Returns the name of the object contained in the specified collection.

Parameters

`single_object_collection`

Specifies the single-object collection that contains the object.

get_path_groups

```
get_path_groups
     [-regexp]
     [-nocase]
     patterns
```

Creates a collection of path groups whose name matches those specified in the pattern list. Assign these path groups to a variable, or pass them to another command. If no objects match the criteria, an empty collection is returned.

Parameters

`-nocase`	Specifies that pattern matching is not case sensitive.
`patterns`	Creates a collection of path groups whose name matches one or more of the specified patterns. Patterns include the * and ? wildcard characters and collections of path groups. If a path group matches multiple patterns, it appears multiple times in the collection.
`-regexp`	Treats the specified patterns a a regular expression pattern. *Default:* Treats the specified pattern as a wild card

get_pins

```
get_pins
     [-hierarchical]
     [-hsc char]
     [-filter expr]
     [-regexp]
     [-nocase]
     [-quiet]
     {patterns | -of_objects object_list}
```

Creates a collection of instance pins whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument to an another command. If no such pin is found, an empty collection is returned.

Parameters

`-filter` *`expr`*	Filters the collection with the specified expression. For any pins that match the pattern, the software evaluates the expression based on the attribute of the pin. If the expression is set to true, the pin is included in the result. You can filter pins using any of the pin properties supported through the `get_property` command.
`-hierarchical`	Specifies that patterns should match hierarchically. If the hierarchical name of a pin at any hierarchical level matches the patterns, it is included in the result.
`-hsc` *`char`*	Specifies the hierarchical delimiter for patterns.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-filter` in order to use this parameter.
`-of_objects` *`object_list`*	Creates a collection of all pins associated with the specified cells or nets. You can specify a list of cells or a list of nets, but you cannot specify a list that contains both cells and nets. You can specify patterns for the cell or net names, a collection of cells or nets, or a combination of patterns and a collection.
`patterns`	Specifies patterns for pin names. Patterns include the * and ? wildcard characters and collections of pins.
`-quiet`	Suppresses all error and warning messages generated when the `get_pins` command is run.

`-regexp`	Treats the specified patterns as a regular expression patterns. **Note:** You must specify `-filter` in order to use this parameter. *Default:* Treats the specified pattern as a wild card

Examples

- The following command returns all instance pins within the hierarchy limited at A:
  ```
  get_pins -hsc @ A@*
  ```
- The following command filters the collection for `pin_direction` pin attribute.
  ```
  get_pins “in*” -filter “@pin_direction == in”
  ```

Related Information

`create_clock`

`get_cells`

`get_ports`

get_ports

```
get_ports
     [-filter expr]
     [-regexp]
     [-nocase]
     [-quiet]
     {patterns | -of_objects object_list}
```

Creates a collection of ports whose name matches the supplied pattern list. Assign this collection to a variable or pass it as an argument to an another command.

Parameters

`-filter` *`expr`*	Filters the collection with the specified expression. For any ports that match the pattern, the software evaluates the expression based on the attribute of the port. If the expression is set to true, the port is included in the result. You can filter ports using any of the port properties supported through the `get_property` command.
`-nocase`	Specifies that pattern matching is not case sensitive. **Note:** You must specify `-filter` in order to use this parameter.
`-of_objects` *`object_list`*	Creates a collection of all ports associated with the specified nets. You can specify patterns for the net names, a collection of nets, or a combination of patterns and a collection.
`patterns`	Specifies patterns for port names. Patterns include the * and ? wildcard characters and collections of ports.The command returns a collection of ports that match the patterns. If no such port is found, an empty collection is returned.
`-quiet`	Suppresses all error and warning messages generated when the `get_ports` command is run.
`-regexp`	Treats the specified patterns as a regular expression patterns. **Note:** You must specify `-filter` in order to use this parameter. *Default:* Treats the specified pattern as a wild card

Example

- The following example returns all ports beginning with the name `inb`:

  ```
  get_ports inb*
  ```

- The following command filters the collection for `port_direction` port attribute.

  ```
  get_ports “mode*” -filter {port_direction == in}
  ```

Related Commands

`all_inputs`

`all_outputs`

`get_clocks`

`get_pins`

get_property

```
get_property
     var_name
     property
     [-clock clock_name]
     [-view view_name]
```

Retrieves the attribute value for the specified object property.

You can query the following:

- Cell Properties
- Clock Properties
- Design Properties
- Library Properties
- Library Cell Properties
- Library Pin Properties
- Library Timing Arc Properties
- Net Properties
- Path Group Properties
- Pin Properties
- Port Properties
- Timing Arc Properties
- Timing Path Properties
- Timing Point Properties

The `get_property` command supports the `report_precision` global variable settings.

Parameters

`-clock clock_name`	Specifies the clock for which to query certain object properties.
property	Specifies the property for which to retrieve the attribute value.

var_name	Specifies a pointer to a collection containing the object. **Note:** If the collection contains more than one object, the `get_property` command returns the property attribute value for the first object in the collection.
`-view` *view_name*	Specifies the analysis view in which to query the object attribute. You can query properties only for the following objects in an analysis view: library cells, library pins, ports, cells, pins, and nets. You can specify this parameter only when the software is in multi-mode multi-corner timing analysis mode.

Cell Properties

`area`	Returns the area of the cell. If the cell is hierarchical, also includes the net area. *Return type:* float
`dbObject`	Returns a pointer to the like object in the database. This pointer can be used with the `dbGet*` commands, and other database commands that take object pointers as arguments.
`early_cell_check_derate_factor`	Returns the derating factor specified through the `set_timing_derate` command with the `-early` parameter. *Return type:* float
`early_clk_cell_derate_factor`	Returns the derating factor specified through the `set_timing_derate` command with the `-early` parameter. *Return type:* float
`early_data_cell_derate_factor`	Returns the derating factor specified through the `set_timing_derate` command with the`-early` parameter. *Return type:* float
`hierarchical_name`	Returns the complete hierarchical name of the cell. *Return type:* string

`is_clock_gating_check`

Returns a value of true if the cell is gated.

Return type: boolean

`is_combinational` Returns a value of `true` if the cell is a combinational cell (not a sequential cell).

Return type: boolean

`is_disable_timing` Returns a value of `true` if the cell's timing has been disabled using `set_disable_timing`.

Return type: boolean

`is_dont_touch` Returns a value of `true` if the cell definition includes `dont_touch:true`.

Return type: boolean

`is_fall_edge_triggered`

Returns a value of `true` if the cell is triggered by the falling edge of the clock.

Return type: boolean

`is_hierarchical` Returns a value of `false` if the cell is a leaf or library cell.

Return type: boolean

`is_integrated_clock_gating_cell`

Returns a value of `true` if the cell is an integrated clock gating cell.

Return type: boolean

`is_negative_level_sensitive`

Returns a value of `true` if the cell is used as a negative level-sensitive sequential element, such as a negative level-sensitive latch.

Return type: boolean

`is_positive_level_sensitive`

Returns a value of `true` if the cell is used as a positive level-sensitive sequential element, such as a positive level-sensitive latch.

Return type: boolean

`is_rise_edge_triggered`

Returns a value of `true` if the cell is triggered by the rising edge of the clock.

Return type: boolean

`is_sequential` Returns a value of `true` if the cell is a latch or flip-flop, or if the cell has sequential timing checks.

Return type: boolean

`is_tristate` Returns a value of `true` if the cell definition includes `three_state:true`.

Return type: boolean

`late_cell_check_derate_factor`

Returns the derating factor for late paths specified through the `set_timing_derate` command with the `-late` parameter.

Return type: float

`late_clk_cell_derate_factor`

Returns the derating factor for late clock paths specified through the `set_timing_derate` command with the `-late` parameter.

Return type: float

`late_data_cell_derate_factor`

Returns the derating factor for late data paths specified through the `set_timing_derate` command with the `-late` parameter.

Return type: float

`name` Returns the leaf-level name of the cell.

Return type: string

`object_type` Returns the object type `cell`.

Return type: string

`pin_count`	Returns the number of pins on the cell. *Return type:* integer
`ref_lib_cell_name`	Returns the library cell name of which the cell is an instantiation. *Return type:* string
`x_coordinate_max`	Returns the maximum x coordinate of the cell. *Return type:* float
`x_coordinate_min`	Returns the minimum x coordinate of the cell. *Return type:* float
`y_coordinate_max`	Returns the maximum y coordinate of the cell. *Return type:* float
`y_coordinate_min`	Returns the minimum y coordinate of the cell. *Return type:* float

Clock Properties

`clock_network_pins`	Returns a collection of all pins on the clock network for the clock. *Return type:* string
`delay_max_fall`	Returns the maximum falling delay value for the clock. *Return type:* float
`delay_max_rise`	Returns the maximum rising delay value for the clock. *Return type:* float
`delay_min_fall`	Returns the minimum falling delay value for the clock. *Return type:* float
`delay_min_rise`	Returns the minimum rising delay value for the clock. *Return type:* float
`hierarchical_name`	Returns the complete hierarchical name of the clock. *Return type:* string

`is_generated` Returns a value of `true` if the clock is a generated clock.

Return type: boolean

`is_propagated_clock` Returns a value of `true` if the clock is in propagated mode.

Return type: boolean

`latency_fall_max` Returns the maximum fall latency for the clock.

Return type: float

`latency_fall_min` Returns the minimum fall latency for the clock.

Return type: float

`latency_rise_max` Returns the maximum rise latency for the clock.

Return type: float

`latency_rise_min` Returns the minimum rise latency for the clock.

Return type: float

`object_type` Returns the object type `clock`.

Return type: string

`period` Returns the clock period.

Return type: float

`source_latency_early_fall_max`

Returns the early source latency for the falling transition at the max corner.

Return type: float

`source_latency_early_fall_min`

Returns the early source latency for the falling transition at the min corner.

Return type: float

`source_latency_early_rise_max`

Returns the early source latency for the rising transition at the max corner.

Return type: float

`source_latency_early_rise_min`

Returns the early source latency for the rising transition at the min corner.

Return type: float

`source_latency_late_fall_max`

Returns the late source latency for the falling transition at the max corner.

Return type: float

`source_latency_late_fall_min`

Returns the late source latency for the falling transition at the min corner.

Return type: float

`source_latency_late_rise_max`

Returns the late source latency for the rising transition at the max corner.

Return type: float

`source_latency_late_rise_min`

Returns the late source latency for the rising transition at the min corner.

Return type: float

`sources` Returns a collection of the clock's source pins or ports.

Return type: collection

`waveform` Returns the rising and falling edge times of the waveform.

Return type: string

Design Properties

`analysis_type` Returns the analysis mode: `single`, `bc_wc`, or `on_chip_variation`.

Return type: string

`capacitance_unit_in_farad`

	Returns the capacitance unit specified in the library. *Return type:* float
`current_unit_in_amp`	Returns the current unit specified in the library. *Return type:* float
`hierarchical_name`	Returns the complete hierarchical name of the design. *Return type:* string
`is_dont_touch`	Returns `true` if the design is specified through the `set_dont_touch` command. *Return type:* boolean
`max_capacitance`	Returns the maximum capacitance value for the design. *Return type:* float
`max_fanout`	Returns the maximum fanout value for the design. *Return type:* float
`max_transition`	Returns the maximum transition design rule limit for the design. *Return type:* float
`min_capacitance`	Returns the minimum capacitance value for the design. *Return type:* float
`object_type`	Returns the object type `design`. *Return type: string*
`operating_condition_max`	Returns the name of the maximum operating condition specified for the design. *Return type:* string
`operating_condition_min`	Returns the name of the minimum operating condition specified for the design. *Return type:* string

`process_max` Returns the process value of the maximum operating condition specified for the design.

Return type: float

`process_min` Returns the process value of the minimum operating condition specified for the design.

Return type: float

`resistance_unit_in_ohm`

Returns the resistance value from the library.

Return type: float

`slew_threshold_percent_fall_high`

Returns the upper slew trip point for falling waveforms, specified as a percent.

Return type: float

`slew_threshold_percent_fall_low`

Returns the lower slew trip point for falling waveforms, specified as a percent.

Return type: float

`slew_threshold_percent_rise_high`

Returns the upper slew trip point for rising waveforms, specified as a percent.

Return type: float

`slew_threshold_percent_rise_low`

Returns the lower slew trip point for rising waveforms, specified as a percent.

Return type: float

`temperature_max` Returns the temperature value of the maximum operating condition for the design.

Return type: float

`temperature_min` Returns the temperature value of the minimum operating condition for the design.

Return type: float

`time_unit_in_second`	Returns the time units of the design, in nanoseconds. *Return type:* float
`tree_type_max`	Returns the value of the `tree_type` attribute for the specified maximum operating condition. *Return type:* string
`tree_type_min`	Returns the value of the `tree_type` attribute for the specified minimum operating condition. *Return type:* string
`voltage_max`	Returns the voltage value of the maximum operating condition for the design. *Return type:* float
`voltage_min`	Returns the voltage value of the minimum operating condition for the design. *Return type:* float
`voltage_unit_in_volt`	Returns the voltage specified in the library for the design. *Return type:* float

Library Properties

`hierarchical_name`	Returns the hierarchical path name for the library. *Return type:* string
`object_type`	Returns the object type `library`. *Return type:* string

Library Cell Properties

`area`	Returns the area of the library cell. *Return Type:* float
`hierarchical_name`	Returns the complete hierarchical name of the library cell (the library name followed by the library cell name). *Return Type:* string

`is_combinational` Returns a value of `true` if the library cell is a combinational cell (not a sequential cell).

Return type: boolean

`is_dont_touch` Returns a value of `true` if the library cell definition includes `dont_touch:true`.

Return Type: boolean

`is_dont_use` Returns a value of `true` if the library cell description includes `dont_use:true`.

Return Type: boolean

`is_fall_edge_triggered`

Returns a value of `true` if the library cell is triggered by the falling edge of the clock.

Return Type: boolean

`is_negative_level_sensitive`

Returns a value of `true` if the library cell is used as a negative level-sensitive latch.

Return type: boolean

`is_positive_level_sensitive`

Returns a value of `true` if the library cell is used as a positive level-sensitive latch.

Return type: boolean

`is_rise_edge_triggered`

Returns a value of `true` if the library cell is triggered by the rising edge of the clock.

Return Type: boolean

`is_sequential` Returns a value of `true` if the library cell is a latch or flip-flop, or if the cell has sequential timing checks.

Return type: boolean

`is_tristate` Returns a value of `true` if the library cell definition includes `three_state:true`.

Return type: boolean

`name`	Returns the leaf-level name of the library cell. *Return Type:* string
`object_type`	Returns the object type `lib_cell`. *Return Type:* string

Library Pin Properties

`capacitance`	Returns the capacitance for the library pin. *Return type:* float
`capacitance_max_fall`	Returns the maximum value of the falling capacitance range. *Return type:* float
`capacitance_max_rise`	Returns the maximum value of the rising capacitance range. *Return type:* float
`capacitance_min_fall`	Returns the minimum value of the falling capacitance range. *Return type:* float
`capacitance_min_rise`	Returns the minimum value of the rising capacitance range. *Return type:* float
`direction`	Returns the direction of the library pin: `in`, `out`, or `inout`. *Return type:* string
`fanout_load`	Returns the fanout load value of the library pin. *Return type:* float
`hierarchical_name`	Returns the hierarchical name of the library pin. *Return type:* string

`is_async` Returns a value of `true` if the library pin is an asynchronous preset pin, or an asynchronous clear pin.

Return type: boolean

`is_clear` Returns a value of `true` if the pin is an asynchronous clear pin.

Return type: boolean

`is_clock` Returns a value of `true` if the library pin definition includes `clock:true`.

Return type: boolean

`is_data` Returns a value of `true` if the library pin is the data pin of a flip-flop.

Return type: boolean

`is_fall_edge_triggered_clock`

Returns a value of `true` if the library pin is the clock pin of a falling edge triggered device.

Return type: boolean

`is_fall_edge_triggered_data`

Returns a value of `true` if the library pin is the data pin of a falling edge triggered device.

Return type: boolean

`is_negative_level_sensitive_clock`

Returns a value of `true` if the library pin is an enable pin of an active low level-sensitive device.

Return type: boolean

`is_negative_level_sensitive_data`

Returns a value of `true` if the pin is a data pin of an active low level-sensitive device.

Return type: boolean

`is_positive_level_sensitive_clock`

Returns a value of `true` if the library pin is an enable pin of an active high level-sensitive device.

Return type: boolean

`is_positive_level_sensitivity_data`

Returns a value of `true` if the pin is a data pin of an active high level-sensitive device.

Return type: boolean

`is_preset` Returns a value of `true` if the pin is a preset pin.

Return type: boolean

`is_rise_edge_triggered_clock`

Returns a value of `true` if the library pin is the clock pin of a rising edge triggered device.

Return type: boolean

`is_rise_edge_triggered_data`

Returns a value of `true` if the library pin is the data pin of a rising edge triggered device.

Return type: boolean

`is_tristate` Returns a value of `true` if the library pin definition includes `three_state:true`.

Return type: boolean

`max_capacitance` Returns the maximum capacitance limit for the library pin.

Return type: float

`max_fanout` Returns the maximum fanout value for the library pin.

Return type: float

`min_capacitance` Returns the minimum capacitance limit for the library pin.

Return type: float

`min_fanout` Returns the minimum fanout value for the library pin.

Return type: float

`name` Returns the leaf-level name of the library cell pin.

Return type: string

`object_type` Returns the object type; in this case, `lib_pin`.

Return type: string

`slew_threshold_percent_fall_high`

Returns the upper slew trip point for falling waveforms, specified as a percent.

Return type: float

`slew_threshold_percent_fall_low`

Returns the lower slew trip point for falling waveforms, specified as a percent.

Return type: float

`slew_threshold_percent_rise_high`

Returns the upper slew trip point for rising waveforms, specified as a percent.

Return type: float

`slew_threshold_percent_rise_low`

Returns the lower slew trip point for rising waveforms, specified as a percent.

Return type: float

Library Timing Arc Properties

`from_lib_pin`	Returns a collection of the source library pins of the library timing arc. *Return type:* collection
`object_type`	Returns the object type `lib_timing_arc`. *Return type:* string
`sdf_cond`	Returns the `sdf_cond` expression specified for the timing library arc in the library. *Return type:* string
`to_lib_pin`	Returns a collection of the sink library pins of the library timing arc. *Return type:* collection

`timing_type`	Returns the type specified for the timing library arc in the library. *Return type:* string
`when`	Returns the `when` condition specified for the library timing arc in the library. *Return type:* string

Net Properties

`capacitance_max`	Returns the total capacitance of the net for maximum conditions. *Return type:* float
`capacitance_min`	Returns the total capacitance of the net for minimum conditions. *Return type:* float
`driver_pins`	Returns a collection of the driver pins of the net. *Return type:* collection
`has_detailed_parasitics`	Returns a value of `true` if the net, or one of its parts, has detailed parasitics associated with it from either RC extraction or SPEF annotation. *Return type:* boolean
`hierarchical_name`	Returns the complete hierarchical name of the net. *Return type:* string
`is_dont_touch`	Returns a value of `true` if a `set_dont_touch` constraint exists for the net. *Return type:* boolean
`load_pins`	Returns a collection of the load pins of the net. *Return type:* collection
`name`	Returns the name of the net. *Return type:* string
`object_type`	Returns the object type `net`. *Return type:* string

`pin_capacitance_max`	Returns the sum of all pin capacitances of the net for maximum conditions. *Return type:* float
`pin_capacitance_min`	Returns the sum of all pin capacitances of the net for minimum conditions. *Return type:* float
`wire_capacitance_max`	Returns the wire capacitance of the net for maximum conditions. *Return type:* float
`wire_capacitance_min`	Returns the wire capacitance of the net for minimum conditions. *Return type:* float

Path Group Properties

`name`	Returns the name of the path group. *Return type:* string
`object_type`	Returns the object type `path_group`. *Return type:* string

Pin Properties

`actual_latency_early_fall_max`

When specified with the `-clock` parameter, returns the worst actual early fall insertion delay in setup mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase.

When specified without the `-clock` parameter, returns the worst actual early fall insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks.

When specified with the `-view` parameter, returns the worst latency for the given view.

Return type: float

`actual_latency_early_fall_min`

When specified with the `-clock` parameter, returns the worst actual early fall insertion delay in hold mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase.

When specified without the `-clock` parameter, returns the worst actual early fall insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks.

When specified with the `-view` parameter, returns the worst latency for the given view.

Return type: float

`actual_latency_early_rise_max`

When specified with the `-clock` parameter, returns the worst actual early rise source insertion delay in setup mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase.

When specified without the `-clock` parameter, returns the worst actual early rise source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks.

When specified with the `-view` parameter, returns the worst latency for the given view.

Return type: float

`actual_latency_early_rise_min`

When specified with the `-clock` parameter, returns the worst actual early rise source insertion delay in hold mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase.

When specified without the `-clock` parameter, returns the worst actual early rise source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks.

When specified with the `-view` parameter, returns the worst latency for the given view.

Return type: float

`actual_latency_late_fall_max`

When specified with the `-clock` parameter, returns the worst actual late fall source insertion delay in setup mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase.

When specified without the `-clock` parameter, returns the worst actual late fall source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks.

When specified with the `-view` parameter, returns the worst latency for the given view.

Return type: float

`actual_latency_late_fall_min`

When specified with the `-clock` parameter, returns the worst actual late fall source insertion delay in hold mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase.

When specified without the `-clock` parameter, returns the worst actual late fall source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks.

When specified with the `-view` parameter, returns the worst latency for the given view.

Return type: float

`actual_latency_late_rise_max`

When specified with the `-clock` parameter, returns the worst actual late rise source insertion delay in setup mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase.

When specified without the `-clock` parameter, returns the worst actual late rise source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks.

When specified with the `-view` parameter, returns the worst latency for the given view.

Return type: float

`actual_latency_late_rise_min`

When specified with the `-clock` parameter, returns the worst actual late rise source insertion delay in hold mode for the specified clock at the given pin. The latency reported is ideal or propagated, depending on the clock phase.

When specified without the `-clock` parameter, returns the worst actual late rise source insertion delay on the pin. If multiple clocks reach the pin, the reported latency is the worst among all the clocks.

When specified with the `-view` parameter, returns the worst latency for the given view.

Return type: float

`arrival_max_fall` Returns the arrival time for the falling transition at the max corner.

Return type: float

`arrival_max_rise` Returns the arrival time for the rising transition at the max corner.

Return type: float

`arrival_min_fall` Returns the arrival time for the falling transition at the min corner.

Return type: float

`arrival_min_rise` Returns the arrival time for the rising transition at the min corner.

Return type: float

`arrival_window` Returns the arrival times for the earliest and latest rising and falling transitions in timing window format.

Return type: string

`capacitance_max_fall`

Returns the maximum value of the falling capacitance range of the corresponding library pin.

Return type: float

`capacitance_max_rise`

Returns the maximum value of the rising capacitance range of the corresponding library pin.

Return type: float

`capacitance_min_fall`

Returns the minimum value of the falling capacitance range of the corresponding library pin.

Return type: float

`capacitance_min_rise`

Returns the minimum value of the rising capacitance range of the corresponding library pin.

Return type: float

`clock_sources`	Returns a collection of the source pins of all the clocks arriving at the pin. *Return type:* collection
`clocks`	Returns a collection of all of the clocks arriving at the pin. *Return type:* collection
`constant_value`	Returns a constant value of `0` or `1`. *Return type:* integer
`dbObject`	Returns a pointer to the like object in the database. This pointer can be used with the `dbGet*` commands, and other database commands that take object pointers as arguments.
`delay_max_fall`	Returns the maximum falling delay. *Return type:* float
`delay_max_rise`	Returns the maximum rising delay. *Return type:* float
`delay_min_fall`	Returns the minimum falling delay. *Return type:* float
`delay_min_rise`	Returns the minimum rising delay. *Return type:* float
`direction`	Returns the direction of the pin: `in`, `out`, or `inout`. *Return type:* string
`fanin`	Returns the number of sources connected with net of the specified pin. *Return type:* integer
`fanout`	Returns the number of sinks connected with the net of the specified pin. *Return type:* integer
`fanout_load`	Returns the fanout load for the pin. *Return type:* float
`hierarchical_name`	Returns the complete hierarchical name of the pin. *Return type:* string

`hold_uncertainty` Returns the most conservative uncertainty of all possible uncertainty assertions associated with the pin.

Return type: float

`is_async` Returns a value of `true` if the pin is an asynchronous clear or preset pin.

Return type: boolean

`is_clear` Returns a value of `true` if the pin is an asynchronous clear pin.

Return type: boolean

`is_clock` Returns a value of `true` if the pin has the Liberty pin attribute: `clock`.

Return type: boolean

`is_clock_gating` Returns a value of `true` if the pin is defined as a pin of a clock gating cell.

Return type: boolean

`is_clock_gating_clock`

Returns a value of `true` if the pin is defined as a clock pin of a clock gating cell.

Return type: boolean

`is_clock_gating_enable`

Returns a value of `true` if the pin corresponds to the enable pin of a clock gating cell.

Return type: boolean

`is_data` Returns a value of `true` if the pin is a data pin (that is, is not a clock pin).

Return type: boolean

`is_disable_timing` Returns a value of `true` if the pin's timing has been disabled.

Return type: boolean

`is_fall_edge_triggered_clock`

Returns a value of `true` if the pin is a clock pin of a flop, and is triggered by the falling edge of a clock.

Return type: boolean

`is_fall_edge_triggered_data`

Returns a value of `true` if the pin corresponds to the data pin of a fall edge triggered device.

Return value: boolean

`is_hierarchical`

Returns a value of `true` if the pin is a pin of a hierarchical module.

Return type: boolean

`is_multiple_clock_fanin_point`

Returns a value of `true` if multiple clock phases converge at the pin.

Return type: boolean

`is_negative_level_sensitive_clock`

Returns a value of `true` if the library pin is an enable pin of an active low level-sensitive device.

Return type: boolean

`is_negative_level_sensitive_data`

Returns a value of `true` if the pin is a data pin of an active low level-sensitive device.

Return type: boolean

`is_pin`

Returns a value of `true`.

The `is_pin` and `is_port` queries are provided for both pin and port objects to provide a quick way of determining the object type of an identifier. `is_port` always returns `false` for pins. `is_pins` always returns `false` for ports.

Return type: boolean

`is_port`

Returns a value of `false`.

The `is_pin` and `is_port` queries are provided for both pin and port objects to provide a quick way of determining the object type of an identifier. `is_port` always returns `false` for pins. `is_pins` always returns `false` for ports.

Return type: boolean

`is_positive_level_sensitive_clock`

Returns a value of `true` if the library pin is an enable pin of an active high level-sensitive device.

Return type: boolean

`is_positive_level_sensitive_data`

Returns a value of `true` if the pin is a data pin of an active high level-sensitive device.

Return type: boolean

`is_preset` Returns a value of `true` if the pin is a preset pin.

Return type: boolean

`is_propagated_clock` Returns a value of `true` if there is an explicit `set_propagated_clock` assertion at the pin.

Return type: boolean

`is_rise_edge_triggered_clock`

Returns a value of `true` if the pin is a clock pin of a flop, and is triggered by the rising edge of a clock.

Return type: boolean

`is_rise_edge_triggered_data`

Returns a value of `true` if the pin corresponds to the data pin of a rise edge-triggered device.

Return type: boolean

`is_tristate` Returns a value of `true` if the pin has the `three_state` attribute in the Liberty timing library.

Return type: boolean

`is_tristate_enable` Returns a value of `true` if the pin is the source pin of timing arcs with either the `three_state_enable` or `three_state_disable` attribute in the Liberty timing library.

Return type: boolean

`is_tristate_output` Returns a value of `true` if the pin corresponds to a three-state output pin.

Return type: boolean

`latency_fall_max`	Returns the maximum fall insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`latency_fall_min`	Returns the minimum fall insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`latency_rise_max`	Returns the maximum rise insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`latency_rise_min`	Returns the minimum rise insertion delay specified by an explicit `set_clock_latency` at the pin. *Return type:* float
`max_capacitance`	Returns the maximum capacitance limit for the pin. *Return type:* float
`max_fanout`	Returns the maximum fanout load that the pin can drive. This value is set using `set_max_fanout` or the `default_max_fanout` library attribute. *Return type:* float
`max_transition`	Returns the maximum transition time limit specified for the pin. *Return type:* float
`min_fanout`	Returns the minimum fanout design rule limit of the corresponding library pin. *Return type:* float
`net_name`	Returns the name of the net connected to the specified pin. *Return type:* string
`object_type`	Returns the object type `pin`. *Return type:* string
`ref_lib_pin_name`	Returns the name of the library pin associated with the instance pin. *Return type:* string

`setup_uncertainty` Returns the most conservative uncertainty of all possible uncertainty assertions associated with the pin.

Return type: float

`slack_max_edge` Returns the edge (rise or fall) of the worst slack-causing path at the specified pin in late mode.

Return type: string

`slack_max_fall` Returns the slack time for the falling transition at the max corner.

Return type: float

`slack_max_rise` Returns the slack time for the rising transition at the max corner.

Return type: float

`slack_min_edge` Returns the edge (rise or fall) of the worst slack-causing path at the specified pin in early mode.

Return type: string

`slack_min_fall` Returns the slack time for the falling transition at the min corner.

Return type: float

`slack_min_rise` Returns the slack time for the rising transition at the min corner.

Return type: float

`slew_max_fall` Returns the slew time for the falling transition at the max corner.

Return type: float

`slew_max_rise` Returns the slew time for the rising transition at the max corner.

Return type: float

`slew_min_fall` Returns the slew time for the falling transition at the min corner.

Return type: float

`slew_min_rise` Returns the slew time for the rising transition at the min corner.

Return type: float

`source_latency_early_fall_max`

Returns the maximum early fall source insertion delay specified by an explicit `set_clock_latency` at the pin.

Return type: float

`source_latency_early_fall_min`

Returns the minimum early fall source insertion delay specified by an explicit `set_clock_latency` at the pin.

Return type: float

`source_latency_early_rise_max`

Returns the maximum early rise source insertion delay specified by an explicit `set_clock_latency` at the pin.

Return type: float

`source_latency_early_rise_min`

Returns the minimum early rise source insertion delay specified by an explicit `set_clock_latency` at the pin.

Return type: float

`source_latency_late_fall_max`

Returns the maximum late fall source insertion delay specified by an explicit `set_clock_latency` at the pin.

Return type: float

`source_latency_late_fall_min`

Returns the minimum late fall source insertion delay specified by an explicit `set_clock_latency` at the pin.

Return type: float

`source_latency_late_rise_max`

Returns the maximum late rise source insertion delay specified by an explicit `set_clock_latency` at the pin.

Return type: float

`source_latency_late_rise_min`

Returns the minimum late rise source insertion delay specified by an explicit `set_clock_latency` at the pin.

Return type: float

`stolen_slack`	Returns the slack stolen at the specified pin. This property can be queried only for pins in late mode. **Note:** The specified pin must be the endpoint of a transparent latch flush arc. *Return type:* float
`x_coordinate`	Returns the x coordinates of the placed pin. *Return type:* float
`y_coordinate`	Returns the y coordinates of the placed pin. *Return type:* float

Port Properties

`arrival_max_fall`	Returns the worst arrival on the port for a falling signal. *Return type:* float
`arrival_max_rise`	Returns the worst arrival on the port for a rising signal. *Return type:* float
`arrival_min_fall`	Returns the best arrival on the port for a falling signal. *Return type:* float
`arrival_min_rise`	Returns the best arrival on the port for a rising signal. *Return type:* float
`arrival_window`	Returns the arrival times for the earliest and latest rising and falling transitions in timing window format. *Return type:* string
`clock_sources`	Returns a collection of the source pins of all the clocks arriving at the port. *Return type:* collection
`clocks`	Returns a collection of all of the clocks arriving at the port. *Return type:* collection
`constant_value`	Returns a constant value of `0` or `1`. *Return type:* integer

`capacitance_max` Returns the maximum pin capacitance for the port.

Return type: float

`capacitance_min` Returns the minimum pin capacitance for the port.

Return type: float

`dbObject` Returns a pointer to the like object in the database. This pointer can be used with the `dbGet*` commands, and other database commands that take object pointers as arguments.

`delay_max_fall` Returns the maximum falling delay value for the port.

Return type: float

`delay_max_rise` Returns the maximum rising delay value for the port.

Return type: float

`delay_min_fall` Returns the minimum falling delay value for the port.

Return type: float

`delay_min_rise` Returns the minimum rising delay value for the port.

Return type: float

`direction` Returns the direction of the port: `in`, `out`, or `inout`.

Return type: string

`drive_resistance_fall_max`

Returns the falling linear drive resistance at the port for the max condition.

Return type: float

`drive_resistance_fall_min`

Returns the falling linear drive resistance at the port for the min condition.

Return type: float

`drive_resistance_rise_max`

Returns the rising linear drive resistance at the port for the max condition.

Return type: float

`drive_resistance_rise_min`

Returns the rising linear drive resistance at the port for the min condition.

Return type: float

`driving_cell_fall_max`

Returns the name of the library cell used to compute max falling delays and transitions at the port.

Return type: string

`driving_cell_fall_min`

Returns the name of the library cell used to compute min falling delays and transitions at the port.

Return type: string

`driving_cell_from_pin_fall_max`

Returns the input pin of the driving cell used in the max fall delay calculation at the port.

Return type: string

`driving_cell_from_pin_fall_min`

Returns the input pin of the driving cell used in the min fall delay calculation at the port.

Return type: string

`driving_cell_from_pin_rise_max`

Returns the input pin of the driving cell used in max rise delay calculation at the port.

Return type: string

`driving_cell_from_pin_rise_min`

Returns the input pin of the driving cell used in min rise delay calculation at the port.

Return type: string

`driving_cell_library_fall_max`

Returns the name of the library used to compute max falling delays and transitions at the port.

Return type: string

`driving_cell_library_fall_min`

Returns the name of the library used to compute min falling delays and transitions at the port.

Return type: string

`driving_cell_library_rise_max`

Returns the name of the library used to compute max rising delays and transitions at the port.

Return type: string

`driving_cell_library_rise_min`

Returns the name of the library used to compute min rising delays and transitions at the port.

Return type: string

`driving_cell_max_fall_itrans_fall`

Returns the falling input transition of the `from_pin` of the driving cell, which is used for the max falling delay calculation at the port.

Return type: float

`driving_cell_max_fall_itrans_rise`

Returns the falling input transition of the `from_pin` of the driving cell, which is used for the max rising delay calculation at the port.

Return type: float

`driving_cell_max_rise_itrans_fall`

Returns the rising input transition of the `from_pin` of the driving cell, which is used for the max falling delay calculation at the port.

Return type: float

`driving_cell_max_rise_itrans_rise`

Returns the rising input transition of the `from_pin` of the driving cell, which is used for the max rising delay calculation at the port.

Return type: float

`driving_cell_min_fall_itrans_fall`

Returns the falling input transition of the `from_pin` of the driving cell, which is used for the min falling delay calculation at the port.

Return type: float

`driving_cell_min_fall_itrans_rise`

Returns the falling input transition of the `from_pin` of the driving cell, which is used for the min rising delay calculation at the port.

Return type: float

`driving_cell_min_rise_itrans_fall`

Returns the rising input transition of the `from_pin` of the driving cell, which is used for the min falling delay calculation at the port.

Return type: float

`driving_cell_min_rise_itrans_rise`

Returns the rising input transition of the `from_pin` of the driving cell, which is used for the min rising delay calculation at the port.

Return type: float

`driving_cell_pin_fall_max`

Returns the name of the output pin used for max fall delay calculation at the port.

Return type: string

`driving_cell_pin_fall_min`

Returns the name of the output pin used for min fall delay calculation at the port.

Return type: string

`driving_cell_pin_rise_max`

Returns the name of the output pin used for max rise delay calculation at the port.

Return type: string

`driving_cell_pin_rise_min`

Returns the name of the output pin used for min rise delay calculation at the port.

Return type: string

`driving_cell_rise_max`

Returns the name of the library cell used to compute max rising delays and transitions at the port.

Return type: string

`driving_cell_rise_min`

Returns the name of the library cell used to compute min rising delays and transitions at the port.

Return type: string

`fanin` Returns the number of sources connected with net of the specified port.

Return type: integer

`fanout` Returns the number of sinks connected with the net of the specified port.

Return type: integer

`fanout_load` Returns the fanout load for ports. (For input ports, the value is the sum of all fanout loads of all connected pins. For output ports, the value is set using `set_fanout_load`.)

Return type: float

`hierarchical_name` Returns the complete hierarchical name of the port.

Return type: string

`hold_uncertainty` Returns the most conservative uncertainty of all possible uncertainty assertions associated with the port.

Return type: float

`input_slew_max_fall` Returns the max falling slew for the input or inout ports. (This value is set with the `set_input_transition` command.)

Return type: float

`input_slew_max_rise`	Returns the max rising slew for the input or inout ports. (This value is set with the `set_input_transition` command.) *Return type:* float
`input_slew_min_fall`	Returns the min falling slew for the input or inout ports. (This value is set with the `set_input_transition` command.) *Return type:* float
`input_slew_min_rise`	Returns the min rising slew for the input or inout ports. (This value is set with the `set_input_transition` command.) *Return type:* float
`is_disable_timing`	Returns a value of `true` if the timing of the port has been disabled using the `set_disable_timing` command. *Return type:* boolean
`is_hierarchical`	Returns a value of `true` if the port is a port of a hierarchical module. *Return type:* boolean
`is_pin`	Returns a value of `false`. The `is_pin` and `is_port` queries are provided for both pin and port objects to provide a quick way of determining the object type of an identifier. `is_port` always returns `false` for pins. `is_pins` always returns `false` for ports. *Return type:* boolean
`is_port`	Returns a value of `true`. The `is_pin` and `is_port` queries are provided for both pin and port objects to provide a quick way of determining the object type of an identifier. `is_port` always returns `false` for pins. `is_pins` always returns `false` for ports. *Return type:* boolean
`is_propagated_clock`	Returns a value of `true` if there is an explicit `set_propagated_clock` assertion at the port. *Return type:* boolean

`latency_fall_max`	Returns the maximum fall insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`latency_fall_min`	Returns the minimum fall insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`latency_rise_max`	Returns the maximum rise insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`latency_rise_min`	Returns the minimum rise insertion delay specified by an explicit `set_clock_latency` at the port. *Return type:* float
`max_fanout`	Returns the maximum fanout load that the port can drive. (This value is set using `set_max_fanout`, or the `default_max_fanout` library attribute.) *Return type:* float
`max_transition`	Returns the maximum transition time limit specified for the port. *Return type:* float
`net_name`	Returns the name of the net connected to the specified port. *Return type:* string
`object_type`	Returns the object type `port`. *Return type:* string
`setup_uncertainty`	Returns the most conservative uncertainty of all possible uncertainty assertions associated with the port. *Return type:* float
`slack_max_edge`	Returns the edge (rise or fall) of the worst slack-causing path at the specified port in late mode. *Return type:* string
`slack_max_fall`	Returns the worst setup slack for a falling signal at the port endpoint. *Return type:* float

`slack_max_rise` Returns the worst setup slack for a rising signal at the port endpoint.

Return type: float

`slack_min_edge` Returns the edge (rise or fall) of the worst slack-causing path at the specified port in early mode.

Return type: string

`slack_min_fall` Returns the worst hold slack for a falling signal at the port endpoint.

Return type: float

`slack_min_rise` Returns the worst hold slack for a rising signal at the port endpoint.

Return type: float

`slew_max_fall` Returns the maximum falling slew from all of the incoming signals on the port.

Return type: float

`slew_max_rise` Returns the maximum rising slew from all of the incoming signals on the port.

Return type: float

`slew_min_fall` Returns the minimum falling slew from all of the incoming signals on the port.

Return type: float

`slew_min_rise` Returns the minimum rising slew from all of the incoming signals on the port.

Return type: float

`source_latency_early_fall_max`

Returns the maximum early fall source insertion delay specified by an explicit `set_clock_latency` at the port.

Return type: float

`source_latency_early_fall_min`

Returns the minimum early fall source insertion delay specified by an explicit `set_clock_latency` at the port.

Return type: float

`source_latency_early_rise_max`

Returns the maximum early rise source insertion delay specified by an explicit `set_clock_latency` at the port.

Return type: float

`source_latency_early_rise_min`

Returns the minimum early rise source insertion delay specified by an explicit `set_clock_latency` at the port.

Return type: float

`source_latency_late_fall_max`

Returns the maximum late fall source insertion delay specified by an explicit `set_clock_latency` at the port.

Return type: float

`source_latency_late_fall_min`

Returns the minimum late fall source insertion delay specified by an explicit `set_clock_latency` at the port.

Return type: float

`source_latency_late_rise_max`

Returns the maximum late rise source insertion delay specified by an explicit `set_clock_latency` at the port.

Return type: float

`source_latency_late_rise_min`

Returns the minimum late fall source insertion delay specified by an explicit `set_clock_latency` at the port.

Return type: float

`x_coordinate` Returns the x coordinates of the placed port.

Return type: float

`y_coordinate` Returns the y coordinates of the placed port.

Return type: float

Timing Arc Properties

`arc_type`	Returns the Liberty `timing_type` attribute of the arc. *Return type:* string
`delay_max_fall`	Returns the fall delay for the max corner. *Return type:* float
`delay_max_rise`	Returns the rise delay for the max corner. *Return type:* float
`delay_min_fall`	Returns the fall delay for the min corner. *Return type:* float
`delay_min_rise`	Returns the rise delay for the min corner. *Return type:* float
`delta_delay_max_fall`	Returns the delta fall delay for the max corner. *Return type:* float
`delta_delay_max_rise`	Returns the delta rise delay for the max corner. *Return type:* float
`delta_delay_min_fall`	Returns the delta fall delay for the min corner. *Return type:* float
`delta_delay_min_rise`	Returns the delta rise delay for the min corner. *Return type:* float
`is_cellarc`	Returns a value of `true` if the arc belongs to a cell. *Return type:* boolean
`mode`	Returns the mode specified for the timing arc in the library. *Return type:* string

`object_type`	Returns the object type `timing_arc`. *Return type:* string
`sdf_cond`	Returns the `sdf_cond` expression specified for the timing arc in the library. *Return type:* string
`sdf_cond_end`	Returns the `sdf_cond_end` expression specified for the timing arc in the library. *Return type:* string
`sdf_cond_start`	Returns the `sdf_cond_start` expression specified for the timing arc in the library. *Return type:* string
`sense`	Returns the sense of the timing arc. *Return type:* string
`sink_pin`	Returns the sink pin of the delay arc, or the reference pin of the check arc. *Return type:* collection
`source_pin`	Returns the source pin of the delay arc, or the signal pin of the check arc. *Return type:* collection
`when`	Returns the `when` condition specified for the timing arc in the library. *Return type:* string

Timing Path Properties

`capture_clock_path`	Returns the capturing clock path of the timing path. *Return type:* collection
`capturing_clock`	Returns a pointer to the capturing clock. *Return type:* collection
`capturing_clock_close_edge_time`	

Returns the closing edge time of the capturing clock.

Return type: float

`capturing_clock_close_edge_type`

Returns the capturing clock edge direction: `rise` or `fall`.

Return type: string

`capturing_clock_is_inverted`

Returns a value of `true` if the phase of the clock at the end point is opposite to the phase of the clock at the source.

Return type: boolean

`capturing_clock_is_propagated`

Returns a value of `true` if the capturing clock is in the propagated mode, at the clock end point.

Return type: boolean

`capturing_clock_latency`

Returns the ideal or propagated latency of the capturing clock.

Return type: float

`capturing_clock_open_edge_type`

Returns the opening edge time of the capturing clock.

Return type: float

`capturing_clock_pin` Returns the name of the capturing clock pin.

Return type: string

`capturing_clock_source_arrival_time`

Returns the capturing clock arrival time at the clock root pin/port of the capturing clock path. This value is the sum of the clock edge time plus any source latency constraint specified at the clock root.

Note: You must specify `report_timing -path_type full_clock` in order to return capturing clock arrival time information.

Return type: float

`capturing_point` Returns a pointer to the end point of the timing path.

Return type: collection

`capturing_point_is_level_sensitive`

Returns a value of `true` if the end point of the data path is level sensitive.

Return type: boolean

`clock_path_end_point`

Returns a pointer to the timing end point.

Return type: collection

`clock_uncertainty` Returns the clock uncertainty of the timing path.

Return type: float

`external_delay` Returns the value of the `set_output_delay` constraint at the end point of the path.

Return type: float

`hold` Returns the hold time at the end point of the path.

Return type: float

`is_transparent_latch`

Returns a value of `true` if the path is a transparent latch path.

Return type: boolean

`launch_clock_path` Returns the launching clock path of the timing path.

Return type: collection

`launching_clock` Returns a pointer to the launching clock.

Return type: collection

`launching_clock_is_inverted`

Returns a value of true if the phase of the clock at the start point is opposite to the phase of the clock at the source.

Return type: boolean

`launching_clock_is_propagated`

Returns a value of `true` if the launching clock is in the propagated mode, at the clock end point.

Return type: boolean

`launching_clock_latency`

Returns the ideal or propagated latency of the launching clock.

Return type: float

`launching_clock_open_edge_time`

Returns the opening edge time of the launching clock.

Return type: float

`launching_clock_open_edge_type`

Returns the launching clock edge direction: `rise` or `fall`.

Return type: string

`launching_clock_source_arrival_time`

Returns the launching clock arrival time at the clock route pin/port of the launching clock path. This value is the sum of the clock edge time plus any source latency constraint specified at the clock root.

Note: You must specify `report_timing -path_type full_clock` in order to return launching clock arrival time information.

Return type: float

`launching_input_delay`

Returns the input delay if the timing path starts from an input or output port. This value is set using `set_input_delay`.

Return type: float

`launching_point` Returns a pointer to the start point of the timing path.

Return type: collection

`launching_point_is_level_sensitive`

Returns a value of `true` if the begin point of the data path is level sensitive.

Return type: boolean

`object_type`	Returns the object type `timing_path`. *Return type:* string
`path_group`	Returns a collection of path groups associated with the timing path. *Return type:* collection
`path_group_name`	Returns the name of the path group for the timing path. *Return type:* string
`path_type`	Returns a value of `max` if the path is for setup, or `min` if the path is for hold. *Return type:* string
`recovery`	Returns the recovery time at the end point of the path. *Return type:* float
`removal`	Returns the removal time at the end point of the path. *Return type:* float
`required_time`	Returns the required time for the timing path. *Return type:* float
`setup`	Returns the setup time at the end point of the path. *Return type:* float
`slack`	Returns the slack of the timing path. *Return type:* float
`time_borrowed`	Returns the amount of time borrowed from the timing end point. *Return type:* float
`time_lent`	Returns the amount of time lent to the timing start point. *Return type:* float
`timing_points`	Returns a collection of the timing points that make up the path. *Return type:* collection

Timing Point Properties

`arrival`	Returns the arrival time at the timing point. *Return type:* float
`object_type`	Returns the object type `timing_point`. *Return type:* string
`pin`	Returns a collection of the pins or ports of the timing point. *Return type:* collection
`slack`	Returns the slack associated with the timing point. *Return type:* float
`slew`	Returns the path-specific slew of the timing point. *Return type:* float
`transition_type`	Returns the transition type of the timing point: `rise` or `fall`. *Return type:* string

Examples

- The following command returns the hierarchical name for the pin contained in the collection referenced by `$pin`:

```
get_property $pin hierarchical_name
```

index_collection

`index_collection` *`base_collection index`*

Returns a collection containing the object that exists at the specified index of the specified object collection.

Parameters

`base_collection`	Specifies the base collection.
`index`	Specifies the index number. You can specify a number from `0` to *`collection_size-1`*.

Examples

- The following command returns the object that exists at the index number 2 in the object collection referenced by `$nets`:

  ```
  index_collection $nets 2
  ```

list_property

`list_property [-type object_type]`

Lists all of the properties associated with the specified object type.

If you do not specify an object type, the `list_property` command lists all of the supported object types, and all of the properties associated with those object types.

Parameters

`-type object_type`	Lists the properties associated with the specified object type. Specify one of the following: `cell`, `clock`, `design`, `lib`, `lib_cell`, `lib_pin`, `lib_timing_arc`, `net`, `path_group`, `pin`, `port`, `timing_arc`, `timing_path`, or `timing_point`.

query_objects

`query_objects` *`collection`* `[-limit` *`value`*`]`

Returns the names of the objects contained inside the specified collection.

Parameters

`collection`	Specifies the object collection for which to return names.
`-limit` *`value`*	Specifies the maximum number of object names to return. Specify a value of `0` to return the names of all of the objects in the collection. *Default:* Returns the first 100 object names in the collection.

Examples

- The following commands create collections of pins and nets, add them together to create a new collection, and return the names of the objects in the collection:

```
set pins [get_pins {i_block1/b1_Sin1 i_block1/b1_CLK1 i_block1/b1_Sout1}]
set nets [get_nets {Sin1 CLK1 Sout1}]

add_to_collection newVar $pins
add_to_collection newVar $cells

query_objects $newVar
```

  The software returns the following information:

```
i_block1/b1_Sin1 i_block1/b1_CLK1 i_block1/b1_Sout1 Sin1 CLK1 Sout1
```

remove_from_collection

`remove_from_collection` *`base_collection object_collection_or_list`*

Creates a new collection by removing the objects specified in one collection from another collection.

Parameters

`base_collection` Specifies the base collection of objects. Objects from the second collection or object list are removed from this collection to create the new collection.

`object_collection_or_list`

Specifies the collection or list of objects that should be removed from the base collection to create the new collection.

Examples

- The following commands create a collection of pins that does not include clock pins:

```
set allP [get_pins *]
set all_pin_no_clock [remove_from_collection $allP [get_pins -filter “is_clock == true”]]
```

You can use the `sizeof_collection` command to see that the `all_pin_no_clock` collection contains fewer objects (that is, no clock pins) than the original `allP` collection:

```
sizeof_collection $allP
50

sizeof_collection $all_pin_no_clock
47
```

report_property

```
report_property
     {collection | list_of_collections}
     [> filename]
```

Reports all of the properties associated with each object in the specified collection.

Parameters

collection | list_of_collections

Specifies the collection, or list of collections for which to report. The collection can contain any of the following object types: `cell`, `clock`, `design`, `lib`, `lib_cell`, `lib_pin`, `lib_timing_arc`, `net`, `path_group`, `pin`, `port`, `timing_arc`, `timing_path`, or `timing_point`.

> filename

Specifies the name of a file in which to save the report.

You can add the `.gz` extension to the file name to generate a compressed report.

Note: The file name must be the last argument in the list.

sizeof_collection

`sizeof_collection` *`collection`*

Returns the total number of objects contained in the specified collection. If the collection contains different types of objects, `sizeof_collection` returns the sum of all the objects of different types in the collection.

Parameters

`collection`	Specifies the collection.

sort_collection

```
sort_collection
     collection
     {property | list_of_properties}
     [-descending]
```

Returns a sorted collection of objects based on the property.

Parameters

collection	Specifies the collection to sort. The collection must be a homogenous collection of any one of the following object types: `cell`, `clock`, `design`, `lib`, `lib_cell`, `lib_pin`, `lib_timing_arc`, `net`, `path_group`, `pin`, `port`, `timing_arc`, `timing_path`, or `timing_point`.
`-descending`	Sorts the objects in the collection by descending order. *Default:* Sorts the collection in ascending order.
property | *list_of_properties*	Sorts the collection by the specified property. You can specify any property associated with the object type. You can specify one or more properties. If you specify more than one property, the software sorts the collection in the order by which you specified the properties.

39

Verify Commands

createMarker

```
createMarker
     -bbox {x1 y1 x2 y2}
     {[-tool toolName] [-layer layerName] [-type typeName]
      [-subtype subtypeName] [-desc description]}
```

Creates markers for violations in the database and imports DRC markers generated by other software applications.

Parameters

`-bbox x1 y1 x2 y2`

Specifies the area of the bounding box of the marker.
Type: Real

`-desc description`

Describes the marker.
Default: `""`
Type: String

`-layer layerName`

Specifies the layer.
Type: String

`-subtype subtypeName`

Specifies the marker subtype.
Default: `Other`
Type: String

`-tool toolName`

Specifies the source software application.
Default: `Other`
Type: String

`-type typeName`

Specifies the marker type.
Default: `Other`
Type: String

Example

The following command generates markers for tool name `Calibre` with type `M1` and subtype `S.1`:

```
createMarker -bbox { 1 2 1 2 } -tool Calibre -type M1 -subtype S.1
```

The violation browser displays the marker in the above example as follows:

```
+ Calibre
    + M1
        + S.1
            ...
```

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Viewing Violation Markers from Assura or Calibre"
  - "Violation Browser Features"
  - "Clearing Violations"

deleteNotchFill

```
deleteNotchFill
```

Deletes all gap, notch, and hole geometries.

- A gap is a minimum spacing violation with two edges without a common adjacent side.
- A notch is a minimum spacing violation with two edges with a common adjacent side.
- A hole is a minimum enclosure violation.

Parameters

None

Related Topics

- ECO Flows appendix to the *Encounter User Guide*

fillNotch

```
fillNotch
     [-area {x1 y1 x2 y2}]
     [-report filename]
     [-useNonDefaultSpacing]
```

Detects and fills notches, gaps, and holes, and corrects acute angle (45 degree) violations.

Creates geometries in `DRCFILL` shapes as special wires attached to each net. Saves the geometries in DEF or GDSII Stream format. In a DEF file, the geometries are saved in the special net section. The software considers the geometries as active design objects.

Parameters

`-area {`*`x1 y1 x2 y2`*`}`

Specifies the area to check and fill.
Type: String

`-report` *`filename`*

Specifies the report file. The file includes the locations, types and number of violations corrected.
Default: *`design`*`.fill.rpt`
Type: String

`-useNonDefaultSpacing`

Honors nondefault spacing rules when filling notches. This behavior matches the behavior of `verifyGeometry`; that is, when this parameter is specified, `fillNotch` honors both default and nondefault spacing rules. If this parameter is not specified, `fillNotch` honors default spacing rules but does not honor nondefault spacing rules.

Related Topics

- ECO Flows appendix to the *Encounter User Guide*

loadDrc

`loadDrc` *`fileName`*

Loads the specified DRC violation marker file. The `loadDrc` command is useful in situations when you want to load DRC violation markers without having to load an entire design.

For example, if during ECO you find that only 25 of the 300 reported violations are actual violations that need to be corrected immediately, you can:

1. Delete the other 250 violations.
2. Save the 25 actual violations in a marker file using the `saveDrc` command.
3. Clear the violations from the design window.
4. Load the marker file containing the saved violation markers using the `loadDrc` command.

You can also use the `loadDrc` command in situations where, when correcting violations, the ECO editor does not make a change in the way that you wanted. Because the editor removed the marker when it made the change, you cannot find the original violation in the design. You can use the `loadDrc` command to load the original marker file, then click on the violation, and zoom to the area in which you were working.

Parameters

`fileName`	Specifies the name of the marker file to load.

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Viewing Violations With the Violation Browser"
  - "Clearing Violations"
- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*

loadViolationReport

```
loadViolationReport
     -type {Assura | Calibre | PVS | Hercules | CDNLitho}
     -filename fileName
```

Loads a violation marker file or a hotspot interface file (HIF) in one of the following formats and creates markers that the Encounter® software can interpret:

- Assura™
- Calibre
- PVS
- Hercules
- Cadence or Calibre hotspot interface format

After you load the file, you can view the markers in the Violation Browser. The Encounter Auto Query feature displays the rule names for the markers.

Parameters

`-filename` *`fileName`*

Specifies the marker file to load.

Assura files must have a `.err` extension. Calibre files must have a `.db` extension.

`-type {Assura | Calibre | PVS | Hercules | CDNLitho}`

Specifies the file type.

The following file types contain violation markers:

- `Assura`
- `Calibre`
- `PVS`
- `Hercules`

The following file types contain hotspot markers, which are used to identify areas susceptible to lithography problems:

- `CDNLitho`

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Viewing Violations With the Violation Browser"
  - "Clearing Violations"
- The Main Window chapter of the *Encounter Menu Reference*
  - "Auto Query"

readTcf

```
readTcf
     tcfFileName
```

Reads a TCF (Toggle Count Format) file from a VCD (Value Change Dump) file.

Parameters

tcfFileName	Specifies the name of the TCF file.

Example

- The following example illustrates an example of creating a TCF file:

```
readTcf design.tcf
```

readVcd

```
readVcd
     vcdFileName
     [-tcfout tcfFileName]
     [-vcdtop topModuleName]
     [-start startTime]
     [-end endTime]
```

Creates a TCF (Toggle Count Format) file from a VCD (Value Change Dump) file.

Parameters

`-end endTime`	Defines the ending time (in nanoseconds) at and before which switching events should be included in the TCF file conversion. *Default*: If you omit this parameter, the power analysis software includes all switching events through the end of the VCD file.
`-start startTime`	Defines the starting time (in nanoseconds) at and after which switching events should be included in the TCF file conversion. *Default*: If you omit this parameter, the power analysis software includes all switching events from the beginning of the VCD file.
`-tcfout tcfFileName`	Specifies the name of the output TCF file.
`vcdFileName`	Specifies the name of the VCD file from which you want to create a TCF file.
`-vcdtop vcdTopModuleName`	Specifies the name of the topmost module in the VCD file at and below which you want to create a report on average activity per net.

Example

- The following example illustrates an example of creating a TCF file:

```
readVcd foo.vcd -tcfout foo.tcf -vcdtop Test/top -start 10 -end 100
```

saveDrc

`saveDrc` *`fileName`*

Saves DRC violation markers in the specified file. The `saveDrc` command is useful for situations where you want to save DRC violation markers without having to save the entire design.

For example, if during ECO you find that only 25 of the 500 reported violations are actual violations that need to be corrected immediately, you can:

1. Delete the other 450 violations.
2. Save the 25 violations in a marker file using the `saveDrc` command.
3. Clear the violations from the design window.
4. Load the marker file containing the saved violation markers using the `loadDrc` command.

Parameters

`fileName`	Saves the DRC violation markers in a file with the specified name.

Command Order

Use this command after markers have been flagged by any of the following commands:

- `verifyGeometry`
- `verifyMetalDensity`
- `verifyProcessAntenna`.

You can also use this command to save Calibre violation markers after a Calibre file has been loaded using the `loadViolationReport` command.

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Viewing Violations With the Violation Browser"
  - "Clearing Violations"

- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*

setNetFreqByTcf

```
setNetFreqByTcf
```

This command should be set when activity from TCF or VCD file needs to be used to capture net frequency for signal EM analysis (verifyACLimit). You must specify TCF or VCD file using readTcf or readVcd command prior to setting this command. In addition, you must use `verifyACLimit-use_db_freq` option to read the toggle rates from activity file.

Parameters

none

Example

- This example reads a TCF file which is used to capture net frequency for signal EM analysis.

```
readTcf design.tcf
setNetFreqByTcf
verifyACLimit -use_db_freq -report signal_em.rpt
```

reportFreqViolation

```
reportFreqViolation
     [-help]
     [-outfile fileName]
     [-selInstFile selInstFileName]
     [-excInstFile excInstFileName]
     [-tableScale float]
     [-detailed]
     [-slew {min | max}]
     [-sort {name | freq | minDiff | maxDiff | minFreq | maxFreq}]
     [-reportFormat integer]
     [-view viewName]
```

Runs electromigration (EM) analysis based on the frequency table defined in the Liberty library and generates a frequency violation report.

You must run the following commands prior to setting the `reportFreqViolation` command:

- `buildTimingGraph`
- `writeTcf` *`fileName`*
- `readTcf` *`fileName`*
- `propagateActivity`
- `SetNetFreqByTcf`

Parameters

`-detailed` — Generates a detailed report containing information for all instances, including those without violations.

`-excInstFile` *`excInstFileName`*

Specifies a file that lists the names of instances to exclude from the frequency violation report.

`-help` — Outputs a brief description that includes type and default information for each `reportFreqViolaltion` parameter.

For a detailed description of the command and all of its parameters, use the man command:
`man reportFreqViolation`.

`-outfile` *`fileName`*

Specifies the report file for the violation data.

`-reportFormat` *`integer`*

Specifies a format of the violation report. The report details vary according to the selected report format. You can specify report formats 0, 1, or 2.
Default: 0

`-selInstFile` *`selInstFileName`*

Specifies a file that lists the names of instances to include in the frequency violation report.

`-slew {min | max}`

Specifies the minimum or maximum slew value.
Default: *`min`*

`-sort {name | freq | minDiff | maxDiff | minFreq | maxFreq}`

Sorts the instance-based frequency data by `name`, `freq`, `minDiff`, `maxDiff`, `minFreq`, and `maxFreq`. By default, the violation report is not sorted.

`name`	Name of a pin or arc.
`freq`	Frequency value of a pin or arc.
`minFreq`	Minimum frequency value of a pin or arc as defined in the frequency table.
`maxFreq`	Maximum frequency value of a pin or arc as defined in the frequency table.
`minDiff`	Difference between `freq` and `minFreq` of a pin or arc.
`maxDiff`	Difference between `freq` and `minFreq` of a pin or arc.

`-view` *`viewName`*

Specifies the view for which to generate the frequency violation report. Use this parameter when working with multi-mode multi-corner (MMMC) designs.

Example

The following command reports instances that violate frequency constraints and lists them in the `freqViol.rpt` report file:

```
reportFreqViolation -outfile freqViol.rpt
```

Instance File Format

The instance file is a text file that contains a list of instances. Each instance name appears in one line in the file. The comments begin with "#". Leading or ending spaces in instance names are ignored.

Report Format

The report formats are described below.

Format 0

The following command generates a report providing details of `minFreq`, `Freq`, `Tran`, `Cap`, `Remark`, and `Arc/Pin` and directs the output to report file `summaryViol.rpt`:

```
reportFreqViolation -outfile summaryViol.rpt -reportFormat 0
```

For example,

```
minFreq    Freq    Tran     Cap        Remark    Arc/Pin
5.985      1.009   0.0356   0.003103             tt0/data/r3/add_27/U1_1_6/Y
```

Format 1

When `reportFreqViolation -reportFormat` is set to `1`, the report provides details as shown in the following example:

```
Status   Remark   freq   minFreq   maxFreq   Arc/Pin
[P]      *        1.009  5.985     5.995     tt0/data/r3/add_27/U1_1_6/Y
```

Format 2

A detailed report is generated when `reportFreqViolation -reportFormat` is set to `2`, as shown in the following example:

```
Status  Remark  freq   minFreq  maxFreq    minTran    maxTran    minCap
maxCap     Arc/Pin
```

```
[P]     *       1.009  5.985    5.995      3.560e-02  3.790e-02  3.103e-03
3.103e-03  tt0/data/r3/add_27/U1_1_6/Y
```

Where:

`Status`	Arc or pin status - `F` (fail), `V` (violation), `P` (pass), or `?` (unknown).
`Remark`	Indicates whether the pin is a clock.
`freq`	Working frequency of the arc or pin.
`minFreq`	Minimum frequency value taken from the frequency table.
`maxFreq`	Maximum frequency value taken from the frequency table.
`minTran`	Minimum slew value.
`maxTran`	Maximum slew value.
`minCap`	Minimum capacitance load.
`maxCap`	Maximum capacitance load.
`Arc/Pin`	Name of the arc or pin.

setSnapGrid

```
setSnapGrid
     -layer {layer_list}
     [-pitch x y]
```

Creates a global snapping grid for wiring (including metal fill shapes and via cuts). The grid defines the snap pitch, which is finer than the LEF `PITCH` for the specified layer. The grid is a multiple of the manufacturing grid.

When you specify this command, the wires, metal fill shapes, and via cuts are snapped to this grid, and the edges of the shapes (for example, wires and vias) lie on the manufacturing grid. The `verifyGeometry` command can check that the shapes are on-grid.

Important

The snap grid is not saved in the Encounter database. To use the snap grid, you must run this command each time you load a design.

For more information, see the commands `addMetalFill`, `addViaFill`, or `verifyGeometry`.

Parameters

`-layer {`*`layer_list`*`}`

Specifies the layers for the snap grid. You can specify metal layers as M1, M2, and so on, and cut layers as V12, V23, and so on.

`-pitch` *`x y`*

Specifies the pitch for the snap grid. The LEF `PITCH` must be a multiple (for example, 2x, 4x, 8x) of the snap grid for the specified layer. For example, if the LEF `PITCH` is 0.2 µ, you can specify a snap grid pitch for both x and y of `0.1` µ.

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Viewing Violations With the Violation Browser"
  - "Clearing Violations"
- Optimizing Metal Density chapter in the *Encounter User Guide*

verifyACLimit

```
verifyACLimit
     [-error value]
     [-ruleFile filename]
     [-nets {netNames} | -selected]
     [-report filename [-detailed]]
     [-scaleIrms value]
     [-toggle value | -use_db_freq]
```

Checks for AC current violations on signal nets.

Calculates the root mean square current (Irms) at the driver output and compares it to the ACCURRENTDENSITY tables in the LEF file that contain the Irms limits for routing layers. Generates an error and attaches a violation marker to a net if the calculated Irms for a net exceeds the ACCURRENTDENSITY Irms limit for a routing layer or width used by the net.

Computes the Irms from the slew rates of the signal, the capacitance of the net, and the toggle-rate frequency as computed by timing analysis commands like `buildTimingGraph` and `check_timing` (and the values can be written out with the `-report` parameter). If there is more than one timing view in use, uses the default setup view (controlled by `set_default_view -setup` *`viewName`* command).

Tip

For increased accuracy on long wires, use SignalStorm® to calculate Irms. In the AC Limit report file, Irms values that are calculated using SignalStorm end with dollar signs ($). To enable SignalStorm calculation, run the following command before running `verifyACLimit`:

```
setDelayCalMode -engine signalStorm
```

For more information, see `setDelayCalMode`.

`verifyACLimit` checks each routing segment separately, eliminating false violations that might be caused by assuming the same current for the entire route. This feature is useful for calculating current at the inputs where it is not as strong due to branching and tapering of wires.

Ignores routing with loops in it, like a clock-mesh, although the initial driver output and wiring Irms that connect to a mesh can be computed if SignalStorm delay calculation is used.

`verifyACLimit` also checks the ACCURRENTDENSITY tables for the following conditions and takes the following actions:

- If there is no table for a routing layer, the software gives a warning and assumes an infinite limit for the layer.

- If PEAK and AVERAGE tables are present, the software ignores them.

Note: This command reports the AC current density in `mA`. The LEF file specifies it in `mA/micron`. To convert the LEF specification, the `verifyACLimit` command reads the value in `mA/micron` and multiplies it by the wire width.

Parameters

`-detailed`

Generates a detailed report containing information for all signal nets, including those without violations.

`-error` *`value`*

Specifies the maximum number of errors to report. The command stops when the maximum value is reached.
Default: `1000`
Range: `0` to `1000000`

`-nets {`*`netNames`*`} | -selected`

Specifies whether to check named or selected nets. If you name only one net, you do not need the braces around the net name. If you specify more than one net, you must separate the net names with a space.
Default: If you do not specify `-nets` or `-selected`, `verifyACLimit` verifies all nets.

`–report` *`filename`*

Specifies the report file for the violation data. The report file includes name of the net, I_{rms}, interpolated `ACCURRENTDENSITY` limit (I_{limit}), layer, width, X and Y location, rise and fall time, effective frequency for the net (F_{eff}), Cnet and Vdd for each net.

`-ruleFile` *`fileName`*

Outputs a file with suggested routing rules for the `fixACLimitViolation` command to follow for widening wires and repairing AC current density violations. Use this parameter after buffer insertion repair with `fixACLimitViolation` is attempted but fails.

The rule file syntax is *`netName ruleName`* where *`ruleName`* is the name of a nondefault rule in the LEF file that is wide enough to avoid AC current violations. For example:

```
/top/net1 rule1
/top/net2 rule2
```

`-scaleIrms` *`value`*

Specifies the scale factor for Irms. The Encounter software multiplies Irms value by the scale factor before checking for violations. The software uses a square wave to approximate current. Specify `1.15` to use a triangular wave.
Default: `1.0`

Note: Useful values for this parameter are from `0.0` to `10.0`. The software accepts values outside of this range for debugging purposes.

`-toggle` *`value`*

Specifies the toggle rate for signal nets. A value of `1.0` means that if a flip-flop is clocked by a 20 Mhz clock, it changes state on 100% of the clock cycles. Therefore the signal has 20 million transitions which is half the number of transitions of the input clock. As a result, the final signal net frequency is 10 Mhz.
Default: `1.0`
Value Range: `0.0` to `1.0`

Note: The `-toggle` parameter and the `-use_db_frequency` parameter are mutually exclusive.

`-use_db_freq`

Uses the database frequency value as the effective frequency per net.

Use the `verifyACLimitSetFreq` command to set the database frequency value on each net in the database using the results from timing analysis. For more information, see verifyACLimitSetFreq on page 2267.

Use the Tcl functions `dbNetFrequency` and `dbSetNetFrequency` to check or modify frequency on any given net before using the `-use_db_freq` parameter in the `verifyACLimit` command. Frequency values are stored in Hertz in the database.

For information on `db*` commands, see the *Encounter Database Access Command Reference*.

Note: The `-toggle` parameter and the `-use_db_frequency` parameter are mutually exclusive.

Optionally read activity from TCF or VCD file using readTcf or readVcd command and then use the setNetFreqByTcf command to calculate the signal frequency.
Example:

```
readTcf design.tcf
setNetFreqByTcf
verifyACLimit -use_db_freq -report signal_em.rpt
```

Example

The following commands check for AC limit violations on all signal nets and generate a report named `aclimit.rpt` for nets with violations. The command uses the setup view specified by the `set_default_view -setup` command.

```
set_default_view -setup view_for_acLimit
extractRC
buildTimingGraph
verifyACLimit -report aclimit.rpt
```

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Verifying AC Limit"

- “Viewing Violations with the Violation Browser”
- “Clearing Violations”

verifyACLimitSetFreq

```
verifyACLimitSetFreq
     [-toggle value]
```

Specifies the frequency on each net in the database. The command uses the results from timing analysis results to calculate the frequency on each net.

Parameters

`-toggle` *`value`*

Specifies the toggle rate for signal nets. A value of `1.0` means that if a flip-flop is clocked by a 20 Mhz clock, it changes state on 100% of the clock cycles. Therefore the signal has 20 million transitions which is half the number of transitions of the input clock. As a result, the final signal net frequency is 10 Mhz.
Default: `1.0`
Value Range: `0.0` to `1.0`

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Verifying AC Limit"
  - "Viewing Violations With the Violation Browser"
  - "Clearing Violations"

verifyConnectivity

```
verifyConnectivity
     [-append]
     [-connectPadSpecialPorts]
     [-connLoop | -geomLoop | -geomConnect]
     [-dividePowerNet]
     [-error integer]
     [-nets {{netNames} | -selected}]
     [-noOpenno_open]
     [-noAntenna]
     [-noUnConnPin]
     [-noUnroutedNet]
     [-noWeakConnect]
     [-rawViolsMark]
     [-report filename]
     [-type {all | special | regular}]
     [-useVirtualConnection]
     [-warning value]
```

Detects conditions such as opens, unconnected wires (geometric antennas), unconnected pins, loops, partial routing, and unrouted nets; generates violation markers in the design window; reports violations. Running this command does not have database impact unless you save the design, which also saves the violation markers.

Parameters

`-append`
: Displays incremental results in the Violation Browser. By default, violation markers are over-written with new results during each `verifyConnectivity` run. When you specify this parameter, violation markers are appended to the violations of the previous `verifyConnectivity` run.

`-connLoop`
: Checks for connectivity loops in regular wires.

  Detects connectivity loops based on the end points of the center line of a regular wire segment or the center of a via.

`-connectPadSpecialPorts`

Treats power and ground pins modelled as “weak connect” as though they are “strong connect” in order to avoid false violations. Specifying this parameter can mask real violations if the I/O has no driving connection between the geometries, as can be the case in libraries where there are several small rings instead of one large ring, due to maximum width constraints.

`-dividePowerNet`

Divides power nets into four subareas for connectivity verification. Use this parameter to decrease memory usage in 32-bit machines with limited memory. This parameter might increase or decrease the run time, depending on the design.

`-error` *`integer`*

Specifies the maximum number of errors to report. The command stops when the maximum value is reached.
Default: `1000`
Range: `0` to `1000000`

`-geomConnect`

Checks for connectivity violations of regular wires. Uses a geometrical model instead of the center-line model. In other words, if the wires overlap at any point, they are considered to be connected—they do not have to connect at the center line.

Use this parameter if you manually change the routing or use a third-party router that does not route using the centerline connection routing technique.

`-geomLoop`

Checks for loop violations of regular nets using a geometrical model. The nets do not have to overlap on the center line. When you specify this parameter, the Encounter software does not perform any other connectivity checks.

Use this parameter if you use a third-party router that does not route using the centerline connection routing technique. In this case, the `-connLoop` parameter might not detect connectivity loop violations.

`-nets {{`*`netNames`*`} | -selected}`

Specifies whether to check named or selected nets. If you specify more than one net, separate the net names with space and surround the list of net names with braces. You can use wildcards (* and ?) when you specify net names.
Default: If you do not specify the `-nets` parameter, `verifyConnectivity` verifies all nets.

`-noAntenna`

Ignores violations due to unconnected wires (also called geometrical antennas or dangling wires).

`-noOpen`

Ignores open violations.

Note: This parameter automatically turns on the `-noUnConnPin` parameter, because open and unconnected pin verification are performed together.

`-noUnConnPin`

Ignores violations due to pins that are not connected to any other objects.

Note: This parameter is specified automatically when you specify the `-noOpen` parameter.

`-noUnroutedNet`

Ignores nets that are not routed.

`-noWeakConnect`

Disables checking for routing to more than one port of the weakly connected pin ports.

`-rawViolsMark`

Displays violation markers for opens as the bounding box of the island. By default, violation markers for opens are displayed as polygons that include all wires, pins, and vias that connect to the island.

`–report` *`filename`*

Specifies the report file for connectivity violation data.

`-type {all | regular | special}`

Specifies the type of wires to verify.
Default: `all`

Note: To check connectivity for whole nets, you do not need to specify a type.

Choose one of the following:

`all`	Checks all wires, including those that have been previously verified.
`regular`	If net names are specified or selected, filters out the special wires in the nets and checks all the remaining (regular) wires.
`special`	If net names are specified or selected, filters out the regular wires in the nets and checks all the remaining (special) wires.

`-useVirtualConnection`

Implies a virtual connection for all bumps and external I/O pins of the same net. Set this parameter to override default behavior of `verifyConnectivity`, in which bumps and external I/O pins of the same net are not considered to be interconnected. It is useful in flip chip designs, where the power bumps are connected outside the chip.

`-warning` *`value`*

Specifies the maximum number of warnings to report.
Default: `50`
Range: `0` to `1000000`
Type: Integer

Example

- The following command verifies connectivity for all nets and generates a maximum of 50 error and 50 warning messages:

  ```
  verifyConnectivity -type all -error 50 -warning 50
  ```

- The below command is same as above except that if net names are specified or selected, filters out the regular wires in the nets and checks all the remaining (special) wires.

  ```
  verifyConnectivity -type special -noAntenna -error 1000 -warning 50
  ```

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Verifying Connectivity"
  - "Viewing Violations With the Violation Browser"
  - "Clearing Violations"
- Signoff in the *Encounter Flat Implementation Flow Guide*

verifyCutDensity

```
verifyCutDensity
     [-area {x1 y1 x2 y2}]
     [-detailed]
     [-globalDensity]
     [-layers {layer1 layer2 ... }]
     [-oversize value]
     [-report fileName]
```

Checks the density of specified cut layers or areas of cut layers, or the cut density of the whole chip.

Parameters

`-area {x1 y1 x2 y2}`

Specifies the coordinates of the area to verify.
Default: Entire area

`-detailed` Generates a detailed cut density report.

`-globalDensity`

Checks the density across the entire design for the specified cut layers and writes the results to the report file.

`-layers {layer1 layer2 …}`

Specifies the cut layers to verify.

Use the following format to specify layer names:

```
{V12 V23}
```

`-oversize value` Specifies an offset value for the area to verify. The value is in user units (not in DBU), and can be positive or negative. A positive value adds to the area that is verified. For example, if the design covers the area from `(0,0)` to `(100,100)` and you specify `-oversize 10`, `verifyCutDensity` checks the area from `(-10,-10)` to `(110,110)`.
Default: `0`

`-report fileName`

Specifies the report file for the cut density violation information.
Default: *designName*`.cutdensity.rpt`

Example

```
verifyCutDensity -layers {V12 V23} \
    -reportfile density.rpt -detailed -globalDensity
```

Related Topics

- Optimizing Metal Density chapter in the *Encounter User Guide*
  - "Overview"
  - "Adding Via Fill"
- "LEF Syntax" chapter in the *LEF/DEF Language Reference*

verifyGeometry

```
verifyGeometry
     [-allowDiffCellViols]
     [-allowPadFillerCellsOverlap]
     [-allowRoutingBlkgPinOverlap]
     [-allowRoutingCellBlkgOverlap]
     [-antenna]
     [-area {x1 y1 x2 y2}]
     [-error integer]
     [-fillToActiveSpacing]
     [-layer layer_list]
     [-layerRange { metalLayerName | bottomMetalLayerName upperMetalLayerName
     }]
     [-maxNonPrefLength value]
     [-minPinArea]
     [-noCornerHaloInfluence]
     [-noImplantCheck]
     [-noInsuffMetalOverlap]
     [-noMaxWidth]
     [-noMergedMGridCheck]
     [-noMinArea]
     [-noMinHole]
     [-noMinimumCut]
     [-noMinStep]
     [-noMinWidth]
     [-noMinSpacing]
     [-noOffMGrid]
     [-noOverlap]
     [-noPinInBlkg]
     [-noRoutingBlkg]
     [-noRoutingBlkgSpacing]
     [-noSameNet]
     [-noShorts]
     [-noViaEnclosure]
     [-noWireExt]
     [-noWireExtAtPin]
     [-offRGrid]
     [-offSnapGrid [-layer layer_list]]
     [-regRoutingOnly]
     [-report filename]
     [-reportAllCells]
     [-reportNetOnlyOffGrid]
     [-sameCellViols]
     [-stackedViasOnRegNets]
     [-useNonDefaultSpacing]
     [-viaOverlap]
     [-warning value]
     [-wireOverlap]
```

Checks width, spacing, and internal geometry of objects and the wiring between them. Creates and saves violation markers in the design database.

Before running this command, ensure the LEF file specifies rules for the objects to verify and the design is routed.

Running verifyGeometry in Multi-Threading Mode

You can add metal fill in multi-threading mode by using the following command before adding the metal fill:

- `setMultiCpuUsage`

For more information, see the Multiple-CPU Processing Commands chapter.

Parameters

`-allowDiffCellViols`

Allows violations between two different cells; does not check object in cells. If you specify this parameter, the following parameters are unset or set automatically:

- Unset:
  - `-sameCellViols`
- Set:
  - `-allowRoutingBlkgPinOverlap`
  - `-allowRoutingCellBlkgOverlap`

`-allowPadFillerCellsOverlap`

Allows overlap violations between overlapping pad filler cells.

`-allowRoutingBlkgPinOverlap`

Allows routing obstructions to overlap pins.

`-allowRoutingCellBlkgOverlap`

Allows overlapping cell and routing obstructions.

`–antenna`

Checks for unconnected wires. Does not report floating fill wires belonging to power and ground nets as antenna violations.

`-area {`*`x1 y1 x2 y2`*`}`

Specifies the coordinates of the area to verify.
Type: Real

`-error` *`integer`*

Specifies the maximum number of errors to report. The software generates a warning message if this number is exceeded.
Default: `1000`

`-fillToActiveSpacing`

Checks for spacing violations between active shapes and DEF `FILLWIRE` shapes.

When this parameter is specified, `verifyGeometry` first uses the value specified with `setMetalFill -activeSpacing`. If `setMetalFill -activeSpacing` is not specified, it uses the LEF `FILLACTIVESPACING` value. If neither `setMetalFill -activeSpacing` nor LEF `FILLACTIVESPACING` is specified, this parameter does not have any effect.

For more information, see `setMetalFill` and the "LEF Syntax" chapter in the *LEF/DEF Language Reference*.

`-layer` *`layer_list`*

Specifies the layers to check for off-grid violations with the `-offSnapGrid` parameter.

Specify a list of layer numbers. The cut layers between the first and last numbers are implied. In the following example, `verifyGeometry` checks the following layers: `1 V12 2 V23 3 V34 4`. It does not check layer `V45`.

```
verifyGeometry -offSnapGrid -layer {1 2 3 4}
```

Default: All layers

`-layerRange {` *`metalLayerName | bottomMetalLayerName upperMetalLayerName`* `}`

Checks a specific metal layer and two cut layers above and below the specified layer. If two metal layer names are specified, `verifyGeometry` checks the cut layer below the lower metal layer and the cut layer above the upper metal layer. You cannot specify three or more metal layer names.

`-maxNonPrefLength` *`value`*

Specifies the maximum length for wires in the nonpreferred routing direction. Reports violations for wires only with lengths longer than the specified value.
Type: Real

`-minPinArea`

Verifies that standard cell pins comply with the minimum area rule (if defined in the LEF file). Flags violations for standard cell pins that do not meet the minimum area rule.

`-noCornerHaloInfluence`

Does not check the influence rule in the halo area surrounding wide wires or objects.

`-noImplantCheck`

Does not check for violations on implant layers. Use this parameter when you run `verifyGeometry` before inserting filler cells.

Note: To skip checking for violations across rows on implant layers, but check for violations within the rows, specify the following variable before invoking `verifyGeometry`:

```
set vgCheckImplantAcrossRows 0
```

`–noInsuffMetalOverlap`

Does not check whether the overlapped of areas of two geometries meet the minimum size requirement for the layer. `verifyGeometry` flags this type of violation as `NSMetal` (for insufficient metal).

`–noMaxWidth`

Does not check for objects with width greater than the maximum width defined in the LEF file.

`-noMergedMGridCheck`

Does not check for objects whose edges are covered by other on-grid shapes. By default, `verifyGeometry` checks and reports violations of these edges.

`–noMinArea`

Does not check for objects smaller than the minimum area specified by the AREA statement in the LAYER (Routing) section of the LEF file.

If AREA is not specified for a layer in the LEF file, and `TOPOFSTACKONLY` via is defined, the `verifyGeometry` command sets the value of AREA for that layer as the area of the bottom rectangle of the `TOPOFSTACKONLY` via.

Note: To skip checking minimum area violations for `IOPINS` only, instead of using `-noMinArea` specify the following variable before invoking `verifyGeometry`:

```
set vgNoMinAreaIOPin 1
```

`-noMinHole`

Does not check for violations where the minimum area of the hole is less than the minimum enclosed area defined in the LEF file.

`-noMinimumCut`

Does not check for violations made by vias that do not have the number of cuts specified by the `MINIMUMCUT` rule. The rule specifies the number of cuts a via must have when it is on a wide wire or pin whose width is greater than width. The rule applies to all vias touching a particular metal layer.

`–noMinSpacing`

Does not check whether the spacing between two geometries violates the minimum spacing rule.

`-noMinStep` Does not check for minimum step violations.

`-noMinWidth`

Does not check for objects smaller than the minimum size specified in the LEF file.

`–noOffMGrid` Does not check for objects whose corners are off the manufacturing grid.

`–noOverlap` Does not check for overlapping components. This parameter also reports overlaps between blockages that belong to different cells. The marker area for the violation encloses the overlap causing the violation.

`-noPinInBlkg`	Does not allow pins inside obstructions of the same component.
`-noRoutingBlkg`	Ignores routing blockages during DRC checking.
`-noRoutingBlkgSpacing`	Does not check for spacing between routing blockages and other geometries, including wires and pins.
`–noSameNet`	Does not check for violations between two geometries belonging to the same net or violations between geometries (such as blockages) that do not have an owning net. These violations can occur when minimum spacing is less than the values defined by the `SAMENET` or `STACK` via rules.
`–noShorts`	Does not check for shorts between two geometries belonging to different nets.
`-noViaEnclosure`	Does not check whether via enclosures meet the specified required x and y enclosure values for the bottom and top metal layers.
`-noWireExtAtPin`	Does not check whether wires are extended at pins.
`-noWireExt`	Does not check for wires that are not extended by the amount specified in the `WIREEXTENSION` definition in the Layer (Routing) section of the LEF file.
`–offRGrid`	Verifies that each geometry covers at least one grid point on the routing grid. This check reports two kinds of violations: non-grid and off-grid. ■ Non-grid violations are reported for blockages, pins, special wires, and special vias that do not enclose a grid-point. ■ Off-grid violations are reported for regular vias whose centers do not lie on-grid and for regular wires whose start and end points do not lie on-grid.

`-offSnapGrid`

Checks for off-grid violations of regular and special wires, placement of LEF vias, and placement and cuts of generated vias. You can limit the layers to check by specifying the `-layer` parameter.

- If `setSnapGrid` is specified for a layer, checks against the snap grid.
- If `setSnapGrid` is not specified for a layer, checks against the manufacturing grid.

For more information, see setSnapGrid on page 2261.

`-regRoutingOnly`

Checks for violations between regular wires (that is, at least one object must be a regular wire), for example, violations between an instance pin and a regular wire. Ignores violations between special geometries.

`-reportAllCells`

Reports violations for all instances. By default, the Encounter software reports violations within instances only once per master, resulting in an overall performance improvement. Specify this parameter to disable this feature.

`-reportNetOnlyOffGrid`

Checks regular and special wires and special and regular vias for off-grid violations.

`-report` *`filename`*

Specifies the report file that contains the violation information.The report contains detailed information that is suitable for debug purposes only. For a concise list of violations, use the Violation Browser report file. For information, see violationBrowserReport on page 2300.
Default: *`designname`*`.geom.rpt`
Type: String

`-sameCellViols`

Checks for violations within a cell.

`-stackedViasOnRegNets`

Checks for stacked vias in regular routing only. Ignores stacked via violations in special routing.

`-useNonDefaultSpacing`

If a net has a nondefault rule, checks the nondefault rule spacing values for every route of the net, including any route segments that may be tapered to default rules in order to reach pins.

`-viaOverlap` — Checks for overlapping vias.

`-warning` *`value`* — Specifies the maximum number of warnings to report. After this number is reached, verification ends.
Default: `50`
Type: Integer

`-wireOverlap` — Checks for overlapping wires.

Parameter `-wireOverlap` is obsolete and has been replaced by checking of LEF `MAXWIDTH` statements on each layer.

Example

The following command verifies the geometry of a design using the following specifications:

- Routing obstructions can overlap pins.
- Overlapping cell and routing obstructions are allowed.
- The command can generate maximum of `1000` error and `50` warning messages.

```
verifyGeometry -allowRoutingBlkgPinOverlap -allowRoutingCellBlkgOverlap
-error 1000 -warning 50
```

Related Topics

- *Encounter User Guide*
  - “Verifying Geometry”
  - “Viewing Violations With the Violation Browser”
  - “Clearing Violations”
  - "Accelerating the Design Process by Using Multiple-CPU Processing"

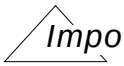
- “Checking for Problems with Cells, Vias, and Pins” in Using the NanoRoute Router chapter
- “Evaluating Violations” in Using the NanoRoute Router chapter

- *LEF/DEF Language Reference*
  - “LEF Syntax”
- *Encounter Flat Implementation Flow Guide*
  - “Route the Design and Run Postroute Optimization”
  - “Signoff”

verifyMetalDensity

```
verifyMetalDensity
     [-report filename]
     [-detailed]
     [-layers {layerNames}]
     [-area {x1 y1 x2 y2}]
     [-saveToDB]
     [-oversize value]
```

Checks the metal density of each routing layer and of macros against values specified in the by `setMetalFill`, by LEF file, or against its own internal default values. Reports density information in the log file.



Important

The `setMetalFill` command is a user-override command that provides constraints at the run time for each session. These constraints have precedence over the LEF/database values. For example, if you specify `setMetalFill -iterationName default`, `verifyMetalDensity` checks against the `default` value.

If neither the LEF file nor `setMetalFill` specifies the window size or density, `verifyMetalDensity` uses internal default values.

Parameters

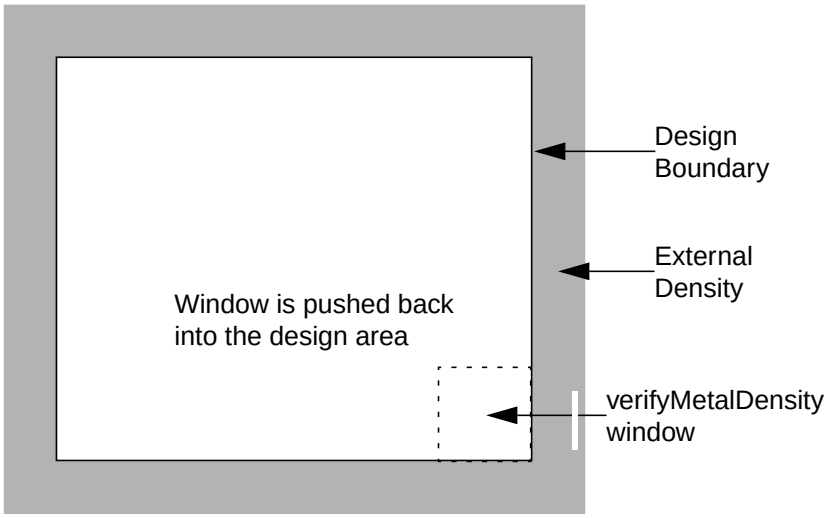
`-area {x1 y1 x2 y2}`

Specifies the coordinates of the area to verify.
Default: Entire area

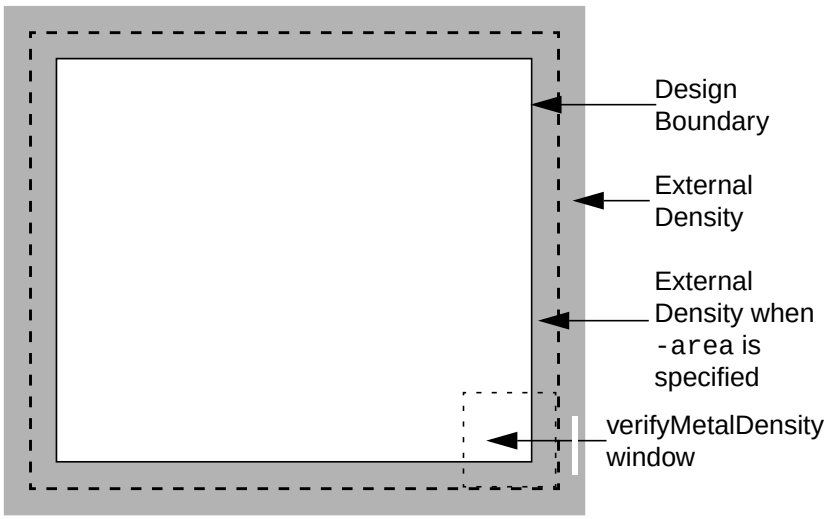
Behavior of verifyMetalDensity -area with external metal density

When metal fill is added to a block design the external metal density provides a density assumption to `verifyMetalDensity`. The external density is used only for density calculation purposes and no fill shapes are added outside of the design area.

When the `-area` parameter is not specified, the external density value is not used. In this case, the `verifyMetalDensity` window is pushed back if it extends outside the design area, as shown in the figure below.



If the `-area` parameter is specified and is larger than the design size then `verifyMetalFill` windows extend outside of the block area and use the external density value (see `setMetalFill` external density value specification) for that area.



If external density is less than the specified preferred density, it is more likely that shapes will be added to the area. If external density is greater than the preferred density it is less likely that shapes will be added (shapes are only added within the design boundary).

Note: Windows outside the design boundary are ignored.

`-detailed`

Generates a detailed metal density report.

`-layers {`*`layerNames`*`}`

Specifies the LEF or Encounter layers to verify. If you specify only one layer name, you do not need the braces around the name.

`-oversize` *`value`*

Specifies an offset value for the area to verify. The value is in user units (not in DBU), and can be positive or negative. A positive value adds to the area that is verified. For example, if the design covers the area from `(0,0)` to `(100,100)` and you specify `-oversize 10`, `verifyMetalDensity` checks the area from `(-10,-10)` to `(110,110)`.
Default: `0`

`-report` *`filename`*

Specifies the report file for the metal density violation information.
Default: *`designName`*`.metaldensity.rpt`

`-saveToDB`

Saves density information to the Encounter database. The `lefOut` command outputs the density information into LEF macro density constructs.

Example

The following command checks metal density and generates a report file named `Test7.density.rpt`:

```
verifyMetalDensity -report Test7.density.rpt
```

The following command checks metal density on layer *M1* in the area specified and generates a detailed report named `test_chip.density.rpt`:

```
verifyMetalDensity -area {-200 -200 1600.0 1600.0} -layers {M1} \
    -reportfile test_chip.density.rpt -detailed
```

In the example above, the layout size is 0 0 1400.0 1400.0. The command considers the density of the area outside of the design as `0`.

Related Topics

- Metal and Via Fill Commands
- "Optimizing Metal Density" chapter in the *Encounter User Guide*
- "LEF Syntax" chapter in the *LEF/DEF Language Reference*
- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Signoff in the *Encounter Flat Implementation Flow Guide*

verifyPowerVia

```
verifyPowerVia
     [-help]
     [-nets {netNames}]
     [-layerRange {bottomLayer topLayer}]
     [-fill]
     [-append]
     [-report file]
```

Checks for metal geometry overlap on power-grid nets and for missing or open vias. The via checks support stacked vias. Violations are highlighted in the layout window and a text report is output with the location of the missing vias.

Parameters

`-append` Specifies that all previous verifyPowerVia settings will be kept.

`-fill` Specifies that metal fill should also be checked.

Default: Metal fill will be ignored.

`-help` Outputs a brief description that includes type and default information for each `verifyPowerVia` parameter.

For a detailed description of the command and all of its parameters, use the `man` command: `man verifyPowerVia`.

`-layerRange {`*`bottomLayer topLayer`*`}`

Optional. The layer range checks for missing stacked vias between only bottom and top layers. It also checks for missing vias between all intermediate layer intersections. You can specify standard Encounter metal layer names, such as `M1`, `M2`, and so on, or metal layer names as defined in the LEF file.

Default: All layers.

If *`bottomLayer`* is specified but *`topLayer`* is not , it assumes one layer up.

`-nets {`*`netNamesnet_names`*`}`

	The name of the power nets that will be checked. A single net or a list of nets enclosed in curly braces({}) or quotes (“ “), can be specified. Wildcards (* and ?) are supported. For example, when you specify `verifyPowerVia -nets` {`v*`}, all PG net names having initial letter as `v` are checked. **Note:** Currently, wildcards are supported for checking PG nets only. *Default*: All power grid nets.
`-report file`	Specifies the name of the output file for the report.

Examples

- Specifies that all vias for all powergrid nets will be checked, except for metal fill and reports it to a file called `power_via.report`:

  ```
  verifyPowerVia-report power_via.report
  ```

- The below command will check for missing stacked vias between M3 and M8 layer intersections on VSS net. In addition it will check for missing vias between M3-M4, M4-M5, M6-M7 and M7-M8 layer intersections. The violations will be reported in powerVia.rpt file and highlighted in GUI.

  ```
  verifyPowerVia -layerRange {M3 M8} \
      -nets VSS -report powerVia.rpt
  ```

verifyProcessAntenna

```
verifyProcessAntenna
     [-detailed]
     [-error value]
     [-leffile filename]
     [-nets {{netNames} | -selected}]
     [-noIOPinDefault]
     [-noMaxFloatingArea]
     [-pgnet]
     [-report filename]
```

Verifies process antenna effect (PAE) and maximum floating area violations.

Before running this command, make sure that process antenna or maximum floating area keywords are specified in the LEF file and the signal nets are routed.

Parameters

`-detailed`	Generates a report containing the specified information for all nets, including those with no violations.
`-error value`	Specifies the maximum number of error messages to report. *Range:* `0` to `1000000` *Default:* `1000`
`-leffile filename`	Specifies the output LEF file. This file contains the process antenna information for the I/O pins of the design.
`-nets {{netNames} | -selected}`	Reports named or selected nets. Select nets by using the `selectNet` command. If you specify one net name, you do not need the braces around the name.
`-noIOPinDefault`	Specifies that the following LEF keywords are not applied to the I/O pins: ■ `ANTENNAINPUTGATEAREA` ■ `ANTENNAINOUTDIFFAREA` ■ `ANTENNAOUTPUTDIFFAREA`

`-noMaxFloatingArea`

Disables check for maximum floating area violations.

`-pgnet`

Checks tie-high and tie-low nets for process antenna violations.

`-report` *`filename`*

Specifies the report file. The file can contain both PAE and maximum floating area violation information.
Default: *`design`*`.antenna.rpt`

Example

The following command verifies PAE violations in the routed design and generates a report file called `Test7.antenna.rpt`:

```
verifyProcessAntenna -report Test7.antenna.rpt
```

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
  - "Verifying Process Antennas"
  - "Viewing Violations With the Violation Browser"
  - "Clearing Violations"
- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - "Repairing Process Antenna Violations"
- *LEF/DEF Language Reference*
  - "Calculating and Fixing Process Antenna Violations" appendix
  - "Defining Routing Layer Properties to Create 45 nm and 65 nm Rules" in the "LEF Syntax" chapter
- Route the Design and Run Postroute Optimization in the *Encounter Flat Implementation Flow Guide*
- Signoff in the *Encounter Flat Implementation Flow Guide*

verifyTracks

verify, verification, tracks, check

```
verifyTracks
```

Checks a predefined set of tracks and reports the following conditions in the Encounter log file:

1. Tracks whose STEP is smaller than the minimum pitch for the designated layer.
2. The percentage of on-track pins and pins that are not accessible (for regular routing).
3. Fully blocked pins and potential off-grid track pins (for special routing).

Before running this command, complete the following steps:

- Load a LEF file with the library and technology information.
- Place the design.

Parameters

None

Related Topics

- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - "Resolving Open Nets"
  - "Diagnosing Problems Using verifyTracks"

verifyWellTap

```
verifyWellTap
     -rule distance
     [-cells {list_of_well_tap_cells}]
     [-fromEdge]
     [-powerDomain powerDomainName]
     [-report filename]
```

Generates violation markers for missing well-tap cells and for well-tap cells that do not meet the rule specified for the distance between well-tap cells and standard cells. Optionally, generates a well-tap violation report.

Marks violations in the area where well-tap placement violates the rule, covering the shared well of abutting rows. If there is no abutting row, marks the violation where it covers only one-half of a row with a violation.

Reports the actual distance between well-tap cells, as well as the rule distance. If there are no well-tap cells in a row and the row that abuts it, or no well-tap cells between the row end and the edge of a block, the software generates a message.

Does not check row areas that are covered by blocks and their halos, or by placement blockages.

The software places well-tap cells with the `addWellTap` command. You can delete well-tap cells with the `deleteFiller` command.

Parameters

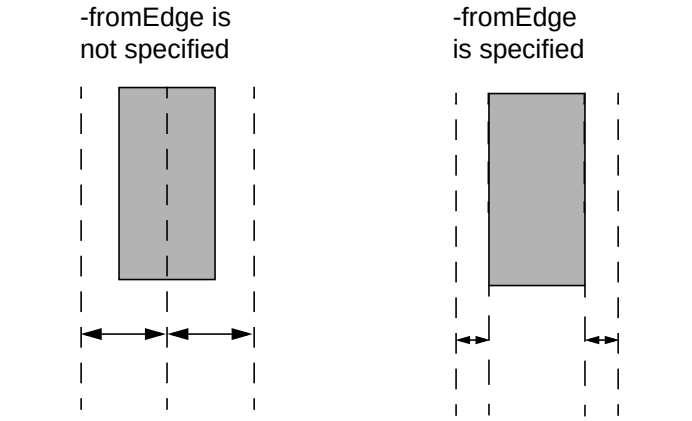
`-cells {list_of_well_tap_cells}`

Specifies the well-tap cells. If not specified, uses cells defined as `CLASS CORE WELLTAP` in the Macro section of the LEF file.

`-fromEdge`

Checks the rule distance from the right and left edges of the well tap cells, instead of from the center line. Use this parameter when the design has well-tap cells of different widths.

If this parameter is not specified, `verifyWellTap` checks the rule distance from the center line of the well-tap cells.



`-powerDomain` *`powerDomainName`*

Limits checking to the specified power domain. If you do not use this parameter, *`verifyWellTap`* checks the entire design.

`-report` *`filename`*

Specifies the report name.
Default: *`topCellName`*`.welltap.rpt`

`-rule` *`distance`*

Specifies the maximum distance between a standard cell and a well-tap cell.

Related Topics

- Placing the Design chapter in the *Encounter User Guide*
  - "Adding Well-Tap Cells"
  - "Deleting Well-Tap Cells"
- Low Power Design chapter in the *Encounter User Guide*
  - "Substrate Biasing"
- ECO Flows appendix to the *Encounter User Guide*

violationBrowser

```
violationBrowser
     [-all]
     [-aclimit]
     [-connectivity]
     [-density]
     [-process_antenna]
     [-geometry]
     [-overlap]
     [-short]
     [-area {x1 y1 x2 y2}]
     [-filter filterName]
     [-filter_mode {AND | OR | NOT}]
     [-search_desc]
     [-no_display_false]
```

Displays a list of violations. The software generates markers for the violations after you use one or more verification commands.

Geometry or Metal Density Violations

Updates geometry and metal density violation markers incrementally during an Encounter session—that is, it displays the markers generated the first time you run the `verifyGeometry` or `verifyMetalDensity` command and adds new markers, or deletes markers, on subsequent runs during the same session. If the software finds violations during a subsequent run that were already found previously, the browser display does not change, as there is no incremental update.

The browser can make incremental changes because `verifyGeometry` and `verifyMetalDensity` can check a small area of the design and update the database. As a result of this behavior, the Encounter software saves the information from the first verification run.

Connectivity, Process Antennas, or AC Limit Violations

Overwrites connectivity, process antenna, and AC limit violation markers if the `verifyConnectivity`, `verifyProcessAntenna`, or `verifyACLimit` commands are run more than once during an Encounter session. These commands are net-based, not area-based, so the browser does not make incremental updates for them. As a result of this behavior, the Encounter software does not keep the information from the first verification run.

Parameters

`-aclimit`

Displays AC limit violations.

`-all`

Displays all the violations that are present in the design.

`-area {`*`x1 y1 x2 y2`*`}`

Specifies the coordinates of the area whose violations to display.

`-connectivity`

Displays connectivity violations.

`-density`

Displays metal density violations.

`-filter` *`filterString`*

Specifies a text string in the violation message (in the bottom portion of the Violation Browser). When you specify a text string, the Violation Browser searches the message of each violation and displays only the violations whose messages match the search conditions.

To specify a list of strings, separate each string with a space.

To specify a literal string with space in it, enclose the string in double quotation marks. For example, you can specify strings such as the following:

- `M5`
- `(600, 500)`
- `M3 (700, 400)`
- `"Pin of Net"`

`-filter_mode {AND | OR | NOT}`

Specifies the condition for searching multiple strings in the violation browser. You cannot filter complex expressions because the `AND`, `OR`, and `NOT` parameters are mutually exclusive.

- The `AND` and `OR` conditions work on single strings or multiple strings.
- The `NOT` condition works only on a single string.

`-geometry` Displays geometry violations.

`-no_display_false`

Prevents the Encounter software from displaying false violations.

`-overlap` Displays overlaps in the design.

`-process_antenna`

Displays process antenna violations.

`-search_desc` Searches for text in the violation message (in the bottom portion of the Violation Browser window) and the violation description (in the top portion of the Violation Browser window).

`-short` Displays short violations.

Examples

The following command displays all violations whose marker title contains `M3`:

```
violationBrowser -all -filter M3
```

The following command displays geometry violations whose marker title contains `M3` or `tdsp`:

```
violationBrowser -geometry -filter {M3 tdsp} -filter_mode OR
```

The following command displays geometry violations whose marker title or marker description field contains `Min` and `0.24` and `M3`:

```
violationBrowser -geometry -filter {Min 0.24 M3} -filter_mode AND -search_desc
```

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*

- ❑ "Viewing Violations With the Violation Browser"
- ❑ "Clearing Violations"

■ Using the NanoRoute Router chapter in the *Encounter User Guide*

- ❑ "Evaluating Violations"

violationBrowserDeleteByArea

```
violationBrowserDeleteByArea
     -area {x1 y1 x2 y2 | all}
```

Deletes makers from a specified area or the entire design area.

Parameters

`-area {x1 y1 x2 y2 | all}`

Specifies the area from which to delete markers.

Specify one of the following:

x1 y1 x2 y2

Specifies a bounding box for the area from which to delete markers.

`all` Deletes markers from the entire design area.

Related Topics

- Verifying Violations chapter in the *Encounter User Guide*
- Using the NanoRoute Router chapter in the *Encounter User Guide*
  - "Evaluating Violations"

violationBrowserReport

```
violationBrowserReport
     [-all]
     [-aclimit]
     [-connectivity]
     [-density]
     [-process_antenna]
     [-geometry]
     [-noDisplayFalse]
     [-overlap]
     [-short]
     [-area {x1 y1 x2 y2}]
     [-report filename]
     [-filter filterString]
     [-filter_mode {AND | OR | NOT}]
     [-search_desc]
```

Reports violations flagged by the verification commands. The report file and violation browser list the violations in the same order.

Parameters

`-aclimit`	Reports AC limit violations.
`-all`	Reports all violations.
`-area {x1 y1 x2 y2}`	Specifies the coordinates of the area for which to report violations.
`-connectivity`	Reports connectivity violations.
`-density`	Reports metal density violations.

`-filter` *`filterString`*

Specifies a text string in the violation message (in the bottom portion of the Violation Browser). When you specify a text string, the Violation Browser searches the message of each violation and displays only the violations whose messages match the search conditions.

To specify a list of strings, separate each string with a space.

To specify a literal string with space in it, enclose the string in double quotation marks. For example, you can specify strings such as the following:

- `M5`
- `(600, 500)`
- `M3 (700, 400)`
- `"Pin of Net"`

`-filter_mode {AND | OR | NOT}`

Specifies the condition for searching multiple strings in the violation browser. You cannot filter complex expressions because the `AND`, `OR`, and `NOT` parameters are mutually exclusive.

- The `AND` and `OR` conditions work on single strings or multiple strings.
- The `NOT` condition works only on a single string.

`-geometry` Reports geometry violations.

`-overlap` Reports overlap violations.

`-no_display_false`

Prohibits the Encounter software from displaying false violations.

`-process_antenna`

Reports process antenna violations.

`-report` *`filename`*

Specifies the report file for the violation information.
Type: String

`-search_desc`	Searches the description field for each marker in addition to the table entries.
`-short`	Reports short violations.

Examples

The following command reports all violations whose marker title contains `M3`:

```
violationBrowserReport -all -filter M3
```

The following command reports geometry violations whose marker title contains `M3` or `tdsp`:

```
violationBrowserReport -geometry -filter {M3 tdsp} -filter_mode OR
```

The following command reports geometry violations whose marker title or marker description field contains `Min` and `0.24` and `M3`:

```
violationBrowserReport -geometry -filter {Min 0.24 M3} -filter_mode AND
-search_desc
```

Related Topics

- "Verifying Violations" chapter in the *Encounter User Guide*

40

What-If Timing Commands

Important

The what-if timing analysis commands do not support the Multi-Mode Multi-Corner (MMMC) feature.

checkWhatIfTiming

```
checkWhatIfTiming
     blackBoxCellName
     [-outfile fileName]
```

Checks that at least one timing arc is defined for every port of a blackbox or blackblob. You use this command to output the status of the timing arcs for all ports to complete the blackbox or blackblob timing arc specifications for the remaining ports.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-outfile *fileName*	Specifies the name of the file to save the information.

Related Topics

- What-If Timing Analysis chapter in the *Encounter User Guide*
  - Performing What-If Timing Analysis
  - Using the What-If Timing Commands

createWhatIfInternalGeneratedClock

```
createWhatIfInternalGeneratedClock
     blackBoxCellName
     -clockName generatedClockName
     -clockPin internalPinName
     -masterPin clockInputPort
     -divideBy integer
     -multiplyBy integer
     [-noCreateInternalPin]
```

Creates an internal pin on a blackbox or a blackblob, and creates a generated clock on this internal pin. The timing graph is updated accordingly.

The following points apply to the usage of this command:

- The internal pin created with this command will be treated as a clock port. This internal pin can, therefore, be used to create check arcs using the setWhatIfTimingCheck command and sequential using the setWhatIfSeqDelay command.

  The latency of the generated clock should include the latency of the master clock. Therefore, the latency of the generated clock should be greater than or equal to the latency of the master clock.

  The delay specified during defining the timing arcs from the internal pins must be greater than or equal to the generated clock latency.

- You can use the generated clock to define timing arcs—these timing arcs are used for timing analysis. Instead of the clock name, the clock pin is referenced while defining the timing arcs.

  Any timing arc related to the internal pin must include the clock latencies defined on both: the internal pin *and* the master pin.

- You can also use this command to specify that a generated clock should be created on an existing internal pin by specifying the `-noCreateInternalPin` parameter.

- If a cell has multiple instances, a generated clock will be defined for each instance of the cell.

- You can set the latency of the generated clock by using the `setWhatIfClockLatency` command. The generated clock appears in the block SDC file generated by the `saveWhatIfTimingModel` command.

- Ensure that the latencies of the generated clock and the master clock are defined before creating any timing arc using the what-if timing analysis commands.

- The internal pin and the generated clock will be treated as what-if assertions and can be deleted with the deleteWhatIfTimingAssertions command.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-clockName *generatedClockName*	Specifies the name of the clock.generated on the internal pin.
-clockPin *internalPinName*	Specifies the name of the pin generated inside the blackbox or blackblob.
-divideBy *integer*	Specifies the frequency division factor. The generated clock period is equal to the master clock period multiplied by this parameter. For example, if the value of this parameter is 2, the generated clock period is two times the master clock period. This parameter takes an integer value.
-masterPin *clockInputPort*	Specifies the source from which the derived clock is generated. **Note:** The source must be an existing clock port. and a clock should be available on this port.
-multiplyBy	Specifies the frequency multiplication factor. The generated clock period is equal to the master clock period divided by this parameter. For example, if the value of this parameter is 2, the generated clock period is half the master clock period. This parameter takes an integer value.
-noCreateInternalPin	Specifies that a generated clock should be created on an existing internal pin. In this case, the -clockPin *internalPinName* parameter should specify the existing internal pin.

Related Topics

- What-If Timing Analysis chapter in the *Encounter User Guide*
  - Using the What-If Timing Commands

Example

The following command created an internal pin named `internalPin1` and a generated clock named `genCLK1` on the blackbox or blackblob named `block3`. The master pin specified is `genCLK1` and the divide-by-frequency specified is 2.

```
createBlackBoxInternalGeneratedClock block3 -clockName genCLK1 -clockPin
internalPin1 -masterPin CLK1-divideBy 2
```

deleteWhatIfTimingAssertions

```
deleteWhatIfTimingAssertions
     [blackBoxCellName]
     [ -port | -from port -to port | -from port | -to port]
     [-assertion_type {comb | setup | hold}]
     [-rising | -falling]
```

Removes the timing arc specifications of the specified blackbox or blackblob. When you use this command, the Encounter software restores the timing arcs of the initial library.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Note: You can delete the internal pin and the generated clock created by the createWhatIfInternalGeneratedClock command by specifying the `-port` parameter of this command. However, the clock and the pin will not be deleted if you use the `-from` *`port`*, `-from` *`port`* `-to` *`port`*, or the `assertion_type` parameters.

Parameters

`blackBoxCellName`	Specifies the name of the blackbox or blackblob cell. If you do not specify this parameter, all timing arc specifications are removed.
`-port` *`portName`*	Specifies the name of the port.
`-from` *`port`* `-to` *`port`*	Specifies that all arcs starting from the port specified with `-from` and ending at the port.specified with `-to`.will be deleted. **Note:** Any constraints specified with the `-assertion_type` parameter and the `-rising` parameter will be applicable. **Note:** The deletion of a timing arc using the `-from` and/or `-to` parameter does not delete the what-if driver assertion on the output ports.
`-from port`	Specifies that all arcs starting from the port specified with `-from` will be deleted. **Note:** Any constraints specified with the `-assertion_type` parameter and the `-rising` parameter will be applicable.

`-to port`	Specifies that all arcs ending at the port.specified with `-to`.will be deleted. **Note:** Any constraints specified with the `-assertion_type` parameter and the `-rising` parameter will be applicable.
`-assertion_type {comb | setup | hold}]`	Specifies one of the following assertion types: ■ setup ■ comb ■ hold
`-rising | -falling`	Specifies whether the timing arcs should be deleted for rising clock transitions or for falling clock transitions. Default: If you do not specify this parameter, timing arcs are deleted for both rising and falling clock transitions.

Related Topics

- What-If Timing Analysis chapter in the *Encounter User Guide*
  - Performing What-If Timing Analysis
  - Using the What-If Timing Commands

getWhatIfTimingAssertions

```
getWhatIfTimingAssertions
     blackBoxCellName
     -port portName
     [-tclList]
```

Displays the what-if timing arc specifications of the specified port in the command window.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-port *portName*	Specifies the name of the port.
-tclList	Generates the specifications as a Tcl list. The Tcl list is useful for integrating what-if timing with custom Tcl functions and customizing specification generation.

Related Topics

- What-If Timing Analysis chapter in the *Encounter User Guide*
  - Using the What-If Timing Commands

saveWhatIfConstraints

```
saveWhatIfConstraints
     [blackBoxCellName]
     [-pt | -dc]
     [-dir dirName]
     [-filePrefix prefix]
```

Saves the what-if timing constraints in PrimeTime or Design Compiler format. You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell. *Default:* If you do not specify a name for the blackbox or blackblob cell, constraints are saved for all blackbox or blackblob cells.
-dir *dirName*	Specifies the name of the directory in which the constraints file will be generated. This directory must reside within the current working directory. *Default:* The constraints file is saved in the current working directory.
-filePrefix *prefix*	Specifies the prefix for the constraints file name. This prefix is added to the name of the blackbox or blackblob cell to generate the name of the constraints file. For example, if you specify the prefix `constraintsFile` and the blackbox or blackblob cell name is `Block1,` the name of the file in which the constraints are saved will be `constraintsFileBlock1.pt` or `constraintsFileBlock1.dc`. *Default:* If you do not specify a prefix, the constraints file is saved as *blackBoxCellName*`.pt` or *blackBoxCellName*`.dc`, where *blackBoxCellName* is the name of the blackbox or blackblob cell.
-pt | -dc	Specifies the constraint format. *Default:* Constraints are saved both in PT and DC format.

Related Topics

- What-If Timing Analysis chapter in the *Encounter User Guide*
  - Performing What-If Timing Analysis
  - Using the What-If Timing Commands

saveWhatIfTimingAssertions

```
saveWhatIfTimingAssertions
     [blackBoxCellName]
     -outfile fileName | -tclList
```

Writes timing arcs to the specified file or Tcl list. The Encounter software saves the final state of the timing arc. The history of all the modifications is not saved. You can source the generated file, regrouping the whole timing arc specifications of the blackbox or blackblob in a new Encounter session.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell. If the name is not specified, all timing arc specifications are saved.
`-outfile` *fileName*	Specifies the name of the output file in which to save the information.
`-tclList`	Generates the timing arcs as a Tcl list. The Tcl list is useful for integrating what-if timing with custom Tcl functions.

Related Topics

- What-If Timing Analysis chapter in the *Encounter User Guide*
  - Performing What-If Timing Analysis
  - Using the What-If Timing Commands

getWhatIfTimingMode

```
getWhatIfTimingMode
     [-quiet]
```

Displays the following information about `setWhatIfTimingMode` parameters in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays values for all of the `setWhatIfTimingMode` mode parameters.

Parameters

`parameter_names`

Displays information for the specified parameters. You can specify one or more parameters.

See `setWhatIfTimingMode` for descriptions of the parameters you can specify.

`-quiet`

Displays the current settings for the specified parameters in Tcl list format only.

If you specify `-quiet` without any parameters, the software displays the current settings of all `setWhatIfTimingMode` parameters in Tcl list format.

Examples

- The following command displays the current settings for the `-model` parameter in Tcl list format onl.y

  ```
  getWhatIfTimingMode -model -quiet
  ```

- The following command displays the current settings for all the `setWhatIfTimingMode` parameters in Tcl list format onl.y

  ```
  getWhatIfTimingMode -quiet
  ```

saveWhatIfTimingModel

```
saveWhatIfTimingModel
     blackBoxCellName
     [-outfile fileName]
```

Saves the specification of the blackbox or blackblob at the top level including the timing arc specifications in the .lib library file format.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-outfile *fileName*	Specifies the name of the file to save the information. *Default: BBoxName*.lib

Related Topics

- What-If Timing Analysis chapter in the *Encounter User Guide*
  - Performing What-If Timing Analysis
  - Using the What-If Timing Commands

setWhatIfClockLatency

```
setWhatIfClockLatency
     blackBoxCellName
     [-clockFrom clockPortName]
     [-init value]
     [-new value]
     [-genConstr value]
```

Specifies the clock insertion delay from a clock input port to register clock input pins within a blackbox or blackblob.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-clockFrom *clockPortName*	Specifies the name of the clock input port. If you do not specify this parameter, the clock insertion delay is specified for all input and bidirectional clock ports.
-init *value*	Specifies the current clock insertion delay value. This value indicates the clock insertion delay used when the software generates the original timing model. This value is used for timing checks or sequential timing arcs. You must specify this value before modifying a timing check or a sequential timing arc. If you modify at least one timing arc, you cannot directly change the clock latency value. You must first delete the timing arc specifications of the blackboxes or blackblobs using the `deleteWhatIfTimingAssertions` command.
-new *value*	Specifies the new clock latency value. This value modifies the clock insertion delay. As the clock insertion delay is included in the timing arc value specified using the `setWhatIfSeqDelay` or `setWhatIfTimingCheck` command, the Encounter software adds the difference between the current and new clock latency value to all timing arcs starting from the specified clock port. Therefore all timing arcs starting from the specified clock port, including the timing arcs that were not modified using the what-if timing commands, are modified.

`-genConstr` *`value`*	Specifies the latency to consider in the what-if timing constraints generation. By default, the value is automatically set to the current clock insertion delay. If you do not modify the latency value, the generated SDC constraints reflect the timing analysis done at the top level. If you set a latency value other than current clock insertion delay, the timing analysis is not impacted.

Related Topics

- What-If Timing Analysis chapter in the *Encounter User Guide*
  - Timing Models Supported for What-If Timing Analysis
  - Using the What-If Timing Commands

setWhatIfClockPort

```
setWhatIfClockPort
     blackBoxCellName
     -port portName
```

Defines a port as a clock port. You can then use the clock port to create timing checks and sequential timing arcs. Therefore, you can do what-if timing analysis without loading the .lib file.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-port *portName*	Specifies the name of the clock port.

setWhatIfCombDelay

```
setWhatIfCombDelay
     blackBoxCellName
     [-from inputPortName]
     [-to outputPortName]
     [-force]
     -delay value
```

Sets the delay for purely combinatorial paths from input ports to the output ports. In the intrinsic timing model mode, the delay corresponds to the delay from the input port to the input of the driver. In the normalized timing model mode, the delay corresponds to the delay from the input port to the output port including the driver delay. If you do not specify the drive type on the output port, a constant timing table is created. In a constant timing table, delay is constant. It does not vary with respect to input slew and output capacitance.

You must load the timing library file containing all pin definitions before using this command.

For more information on the timing models, see the “What-If Timing Analysis” chapter in the *Encounter User Guide*.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-delay *value*	Specifies the delay value.
-force	Creates timing arcs even if they are not defined in the timing library. If you do not specify this parameter, and the combinatorial arc specification is not defined in the initial timing library, then the timing arc is not created.
-from *inputPortname*	Specifies the name of the input port. If you do specify this parameter, the software uses the delay value for all data input and bidirectional ports.
-to *outputPortname*	Specifies the name of the output port. If you do not specify this parameter, the software uses the delay value for all output and bidirectional ports.

setWhatIfDriveType

```
setWhatIfDriveType
     blackBoxCellName
     [-port outputPortName]
     [-lib libName]
     -cell cellName
     [-slew value [-outputCap value]]
     [-inputPin libraryInputPin]
     [-outputPin libraryOutputPin]
     [-outputCap value]
```

Sets the output and bidirectional ports that have the specified type of cell within the blackbox or blackblob as drivers on the interface nets. You can not specify two different drivers for two timing arcs ending at the same output port. The Encounter software stores the drive type specification, and uses it to create a timing arc after you specify the delay.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Note: The timing sense of the driver is taken into account in the combinatorial what-if timing arc description—while applying the drive type, the timing sense of the combinatorial arc is replaced by the timing sense of the driver's timing arc. For sequential arcs, the timing sense is always set to `non_unate`.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
`-cell` *cellName*	Specifies the name of the cell. If there are two cells with the same name, the software uses the first one found. You can specify a complex cell (with multiple inputs and/or outputs) with this parameter.
`-lib` *libName*	Specifies the name of the timing library.

`-outputCap` *`value`*

The value for this parameter is set as follows:

- If the value of the -context parameter of the `setWhatIfTimingMode` is set to `in`, the output capacitance value is equal to the real capacitance. The `-outputCap` parameter of the `setWhatIfDriveType` command is not applicable in this case.
- If the value of the -context parameter of the setWhatIfTimingMode is set to `out`:
  - If the `-outputCap` parameter is specified with the `setWhatIfDriveType` command, the output capacitance is set to this value.
  - If the `-outputCap` parameter is *not* specified with the `setWhatIfDriveType` command, the outputCap value as specified with the `-outputCap` parameter of the `setWhatIfTimingMode` is used as the output capacitance for the `setWhatIfDriveType` command.

Important

You cannot use this parameter in the intrinsic timing model mode.

`-port` *`outputPortName`*

Specifies the name of the output port. If you do not specify a port name, the software sets all output and bidirectional ports as drivers on the interface nets. Each time you modify the driver on an output port of the blackbox or blackblob, the software recalculates all timing arcs attached to the output port with the new driver specification.

`-slew` *`value`*

Specifies the slew at the input of the driver cell. If you specify the slew value, the software creates a one-dimensional timing table depending on the output capacitance.
If you do not specify the slew value, the software creates a two-dimensional timing table with input slew and output capacitance values. In this case, the software assumes that there is no slew variation between the input port of the blackbox or blackblob, and the driver input pin.
You must use this parameter in the normalized timing model mode. This parameter is not available in the intrinsic timing model mode

`-inputPin` *`LibraryInputPin`*

Specifies the input or inout pin of the cell (specified by the `-cell` parameter) whose source arcs will be used for setting the drive type on the port.

Default: If this parameter is not specified, the Encounter software selects an input or inout pin automatically.

`-outputPin` *`LibraryOutputPin`*

Specifies the output or inout pin of the library whose source arcs will be used for setting the drive type on the port.

Default: If this parameter is not specified, the Encounter software selects an output or inout pin automatically.

setWhatIfLoadType

```
setWhatIfLoadType
     blackBoxCellName
     [-port inputPortName]
     [-lib libName]
     -cell libraryCellName
     [-inputPin libraryInputPin]
     [-multiply_by value]
```

Specifies that the load on any input or any inout what-if port should be the same as that of the specified library input or inout pin.

Parameters

blackBoxCellName Specifies the name of the blackbox or blackblob cell.

-cell library*CellName*

Specifies the name of the cell from which the load value will be picked.

-inputPin *libraryInputPin*

The library cell input or inout pin whose source arcs will be used for setting the load on the port.

Default: If this parameter is not specified, the Encounter software selects an input or inout pin automatically.

-lib *libName* Specifies the name of the timing library from which the specified cell will be picked.

Default: The library is selected automatically based on the current context.

-multiply_by *value* Specifies the load value multiplication factor. The applied load on the blackbox or blackblob is equal to the load of the specified cell pin multiplied by this factor.

This parameter takes an integer or floating point value.

Default: If no value is specified, the multiplication factor is set to 1.

-port *inputPortName*

Specifies the input or inout port of the blackbox or blackblob.

Default: If no port is specified, the load is applied to all ports.

Examples

The following command sets the load on the `input_Port1` of blackbox `Blackbox1` to the same as that on input pin `Pin_A1` of cell bufx1 of library `library_DelayLogic`.

```
setWhatIfLoadType Blackbox1 -port input_Port1 -lib library_DelayLogic -cell bufx1
-inputPin Pin_A1
```

setWhatIfPortParameters

```
setWhatIfPortParameters
     blackBoxCellName
     [-port portName]
     [-cap capacitanceValue]
     [-max_cap maxCapacitanceValue]
     [-max_trans maxTransitionValue]
     [-max_fanout maxFanoutValue]
```

Sets the following values on the what-if cell ports:

- Capacitance
- Maximum capacitance
- Maximum transition
- Maximum fanout

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

`blackBoxCellName`
: Specifies the name of the blackbox or blackblob cell.

`-port portname`
: Specifies the name of the clock port on which to set the value(s).

Default: By default, the specified parameters are applied to all ports in the what-if timing analysis model.

Note: The value specified for a particular port override the global values. For example, consider the following use case scenario for a cell `b_cell`:

1. You specify a maximum fanout value for all ports to 20 (without specifying any port with the command):

```
setWhatIfPortParameters b_cell -max_fanout 20
```

2. You then run the command again on port `out1` with a maximum fanout value of 5.

```
setWhatIfPortParameters b_cell -max_fanout 5 -port out1
```

In this case, the maximum fanout for port `out1` will be set to 5.

`-cap` *`capacitanceValue`*

Specifies the capacitance value in picoFarad. This is the capacitance within the blackbox or the blackblob, on the specified I/O port.

The value can be in decimal, for example, `0.75`.

`-max_cap` *`maxCapacitanceValue`*

Specifies the maximum capacitance value. This is the maximum value of capacitance permitted on the net going through the I/O port.

`-max_fanout` *`maxfanOutValue`*

Specifies the maximum fanout value. The value should be an integer, for example 20.

`-max_trans` *`maxTransitionValue`*

Specifies the maximum transition value in nanoseconds. The value can be in decimal, for example, 5.3.

Examples

- The following command sets the maximum capacitance value to `1.0` and the maximum fanout value to `20` for the port `a1_outport1` of the blackbox or blackblob cell named `blockCell2A`.

  ```
  setWhatIfPortParameters blockCell2A -port a1_outport1 -max_fanout 20 -max_cap 1.0
  ```

- The following command is similar to the previous command, except that it sets the values for all ports (as no port is specified with the command)

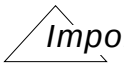
  ```
  setWhatIfPortParameters blockCell2A -max_fanout 20 -max_cap 1.0
  ```

- The following set of commands set the values as follows
  - the maximum maximum fanout value to `20` for all the ports except port `a1_outport1`
  - the maximum fanout value to 5 and the maximum capacitance value to 1.3 for port `a1_outport1`

  ```
  setWhatIfPortParameters blockCell2A -max_fanout 20
  setWhatIfPortParameters blockCell2A -max_fanout 5 -max_cap 1.3 -port a1_outport1
  ```

setWhatIfPortPriority

```
setWhatIfPortPriority
     [-driveCell | -portParam]
```



Important

Command `setWhatIfPortPriority` is obsolete and has been replaced by setWhatIfTimingMode.

On output ports, parameters such as capacitance value, maximum capacitance values, maximum transition value, or the maximum fanout value can come from the driver (`setWhatIfDriveType` command) or they can be set through the `setWhatIfPortParameters` command.

The `setWhatIfPortPriority` command defines which of these values will take precedence in case of a conflict.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Note: This feature is available only in the CTE mode.

Parameters

`-driveCell`	Specifies that the values set through the setWhatIfDriveType command will take precedence
`-portParam`	Specifies that the values set through the setWhatIfPortParameters command will take precedence.

The default value is `-portParam`.

setWhatIfSeqDelay

```
setWhatIfSeqDelay
     blackBoxCellName
     [-clockFrom clockPortName]
     [-to outputPortName]
     [-clockRise | -clockFall]
     [-force]
     -delay value
```

Sets the delay from a clock input port to an output port of the blackbox or blackblob so that the delay includes the clock latency. In the intrinsic timing model mode, the delay is from the clock input port to the input of the driver. In the normalized timing model mode, the delay is from the clock input port to the data output port including the driver delay. For more information on the timing models, see the “What-If Timing Analysis” chapter in the *Encounter User Guide*.

To create a new sequential timing arc which does not already exist in the timing model, the clock port that you specify in this command must be already referenced as a clock pin in the timing model.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O’s in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-clockFrom *clockPortname*	Specifies the name of the clock port. If you do not specify this parameter, the delay is set for all input and bidirectional clock ports.
-clockRise | -clockFall	Specifies the transition on the clock input relevant to the sequential delay arc definition. By default, the software uses a rising clock transition.
-delay *value*	Specifies a value for the delay.

`-force`

Creates a timing arc even if it is not defined in the timing library. If you do not specify this parameter, and the sequential arc specification is not defined in the initial timing library, then the timing arc is not created.

`-to` *`outputPortName`*

Specifies the name of the output port. If you do not specify a port name, the software sets the delay for all output and bidirectional data ports.

setWhatIfTimingCheck

```
setWhatIfTimingCheck
     blackBoxCellName
     [-setup | -hold]
     [-clockFrom clockPortName]
     [-to inputPortName]
     [-clockRise | -clockFall]
     [-force]
     -delay value
```

Sets the delay from the clock input port to the data input port. The delay value contains the clock latency.

To create a new timing check arc which does not already exist in the timing model, the clock port that you specify in this command must be already referenced as a clock pin in the timing model.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

blackBoxCellName	Specifies the name of the blackbox or blackblob cell.
-clockFrom *clockPortname*	Specifies the name of the clock port. If you do not specify this parameter, the timing check is set for all input and bidirectional clock ports.
-clockRise | -clockFall	Specifies which transition on the clock input is relevant for the sequential delay arc definition. By default, the software uses a rising clock transition.
-delay *value*	Specifies a value for the delay.
-force	Creates a timing arc even if it is not defined in the timing library. If you do not specify this parameter, and the sequential arc specification is not defined in the initial timing library, then the timing arc is not created.
-setup | -hold	Specifies the setup or hold time for the timing check.
-to *inputPortname*	

Specifies the name of the output port. If you do not specify a port name, the software sets the timing check for all output and bidirectional data ports.

setWhatIfTimingMode

```
setWhatIfTimingMode
     [-help]
     [-reset]
     [-context {in | out}]
     [-defaultOutputCap value]
     [-model {intrinsic | normalized}]
     [-portPriority {cellType | portParam}]
```

Sets the timing model mode for all blackboxes or blackblobs of the design. You can select only one mode for all blackboxes or blackblobs. For more information on the timing models, see "What-If Timing Analysis," in the *Encounter User Guide*.

If you modify at least one timing arc, you cannot directly change the mode for the blackboxes or blackblobs. You must first delete the timing arc specifications of the blackboxes or blackblobs using the deleteWhatIfTimingAssertions command.

You can use this command after the module definition of the blackbox or the blackblob is available with the I/O's in the Verilog netlist.

Parameters

`-context [{in | out}]`

Specifies whether the timing mode description is `in-context` or `out-context`. The context refers to the chip-level context.

- In-context: The timing mode description is created once the chip environment is ready, that is, after placement and routing. In this case, the Encounter software takes into consideration the capacitance of the design.
- Out-context: The timing model is described outside the chip environment and can be described before floorplanning. You provide the value of the timing arc for a given capacitance.

The value of the `-outputCap` parameter of the setWhatIfDriveType command is set based on whether the context is `in` or `out`.

Important

The `-context` parameter can be used only in the normalized mode.

The value of the `-context` parameter is frozen once a timing arc is defined. This is done to avoid having an incorrect timing assertion file later in the flow.

Default: The default value is `out`.

`-defaultOutputCap [`*`value`*`]`

Specifies the value of the default output capacitance on the port.

Important

The `-defaultOutputCap` parameter can be used only in the normalized mode.

The value of the `-defaultOutputcap` parameter is frozen once a timing arc is defined. This is done to avoid having an incorrect timing assertion file later in the flow.

The unit is pico farads.

Default: The default value is 0.

`-help`

Outputs a brief description that includes type and default information for each `setWhatIfTimingMode` parameter.

For a detailed description of the command and all of its parameters, use the man command: `man setWhatIfTimingMode`.

`-model {intrinsic | normalized}`

Specifies whether the timing model mode is *intrinsic* or *normalized*.

Default: The default mode is *normalized*.

`-portPriority {cellType | portParam}`

On output ports, parameters such as capacitance value, maximum capacitance values, maximum transition value, or the maximum fanout value can come from the driver (`setWhatIfDriveType` command) or they can be set through the `setWhatIfPortParameters` command. You can use the `-portPriority` parameter to specify which of these values take precedence in case of a conflict.

- The value `cellType` specifies that the parameter values set through the `setWhatIfDriveType` command take precedence.
- The value `portParam` specifies that the parameter values set through the `setWhatIfPortParameters` command take precedence.

Important

The value of the portPriority parameter is frozen once a what-if timing arc has been defined. This avoids inconsistencies between he timing analysis files and the files that you can generate (such as whatif assertions, timing model, or SDC) if you change the port priority subsequently.

Default: The default value is `portParam.`

`-reset`	Resets parameters to their default values. The `-reset` parameter *must* be the first parameter specified. If you specify `-reset` by itself, the software resets all `setWhatIfTimingMode` parameters to their default values. If you specify parameters after `-reset`, the software resets only those parameters to their default values.

Examples

- The following command sets the timing model mode to intrinsic and specifies that in case of a conflict, the parameter values set through the `setWhatIfDriveType` command takes precedence over the parameter values set through the `setWhatIfPortParameters` command. Because the `-defaultOutputCap` parameter and the -context parameter were not specified, the output capacitance is set to `0`, and the context is set to `out`.

  ```
  setWhatIfTimingMode -model intrinsic -portPriority cellType
  ```

- The following command resets the value of the `-model` and the `-portPriority` parameters. Note that the `-reset` parameter is the first parameter in the command.

  ```
  setWhatIfTimingMode -reset -model -portPriority
  ```

- The following command resets the value of all the `setWhatIfTimingMode` parameters

  ```
  setWhatIfTimingMode -reset
  ```

41

Wire Edit Commands

convertNetToSNet

`convertNetToSNet [-nets netNames] [-exclude excluded_netNames]`

Converts regular nets to geometrically equivalent special nets, so you can push down the routing information. You can convert all nets, named nets, or all nets except those you specify.

After routing, you can convert the nets back to regular nets by using the `convertSNetToNet` command. For more information, see "convertSNetToNet" on page 2341.

Note: Use this command to convert the physical nets, after routing. To convert logical nets, use `setNets`.

Parameters

Note: If you do not specify any parameters, this command converts all signal nets to special nets.

`-exclude excluded_netNames`

Specifies nets you do not want to convert. You can use wildcards.

- If a name contains a special character, enclose it in braces ({}).
- If you specify multiple net names, enclose the names in braces ({}).

`-nets netNames`

Specifies the nets to convert. You can use wildcards.

- If a name contains a special character, enclose it in braces ({}).
- If you specify multiple net names, enclose the names in braces ({}).

Note: For backward compatibility, this command also supports the following syntax for nets to convert:

`convertNetToSNet netName [netName2 netName3 …]`

Examples

- The following command converts regular nets named `clk*` to special nets:

  ```
  convertNetToSNet -nets clk*
  ```

- The following command converts all regular nets, except those named `clk1*`, to special nets:

  ```
  convertNetToSNet -exclude clk1*
  ```

- The following command converts regular nets named `clk*` to special nets, except nets named `clk1*`:

  ```
  convertNetToSNet -nets clk* -exclude clk1*
  ```

- The following command converts all regular nets named `clk*`, `a2`, and `s2[1]`, except for nets `clk1` and `clk2`, to special nets:

  ```
  convertNetToSNet -nets {clk* a2 s2[1]} -exclude {clk1 clk2}
  ```

convertSNetToNet

`convertSNetToNet [-nets` *`netNames`*`] [-exclude` *`excluded_netNames`*`]`

Converts special nets to geometrically equivalent regular nets. You can convert all nets, named nets, or all nets except those you specify.

Use this command after converting nets from regular to special, after you push down the routing information to partitions. Do not touch the routing, or change attributes or the LEF file.

After converting the nets back to regular, the associated partition pin is still marked `USE SPECIAL`. The special pin should not cause a problem during RC extraction, timing analysis, or the chip assembly flow.

For information on converting regular nets to special nets, see “convertNetToSNet” on page 2339.

Note: Use this command to convert the physical nets, after routing. To convert logical nets, use `setNets`.

Parameters

Note: If you do not specify any parameters, this command converts all signal nets to special nets.

`-exclude` *`excluded_netNames`*

Specifies nets you do not want to convert. You can use wildcards.

- If a name contains a special character, enclose it in braces (`{}`).
- If you specify multiple net names, enclose the names in braces (`{}`).

`-nets` *`netNames`*

Specifies the nets to convert. You can use wildcards.

- If a name contains a special character, enclose it in braces (`{}`).
- If you specify multiple net names, enclose the names in braces (`{}`).

Note: For backward compatibility, this command also supports the following syntax for

nets to convert:

```
convertSNetToNet netName [netName2 netName3 …]
```

Examples

- The following command converts special nets named `clk*` to regular nets:

  ```
  convertSNetToNet -nets clk*
  ```

- The following command converts all special nets, except those named `clk1*`, to regular nets:

  ```
  convertSNetToNet -exclude clk1*
  ```

- The following command converts special nets named `clk*` to regular nets, except nets named `clk1*`:

  ```
  convertNetToSNet -nets clk* -exclude clk1*
  ```

- The following command converts all special nets named `clk*`, `a2`, and `s2[1]`, except for nets `clk1` and `clk2`, to regular nets:

  ```
  convertNetToSNet -nets {clk* a2 s2[1]} -exclude {clk1 clk2}
  ```

editAddFillet

```
editAddFillet
```

Adds a tear-drop fillet between a wire and bump connection. A teardrop fillet is used to reinforce the mechanical strength of the connection to the bump. This applies to all bumps in the design. If the connected wire is too short to create the fillet, a marker is created to indicate the failure.

editAddPoly

`editAddPoly` *x y*

Creates a point that is part of a polygon that starts or ends at the specified coordinates.

Parameters

x y	Specify the x and y coordinates for the startpoint or endpoint of the wire segment.

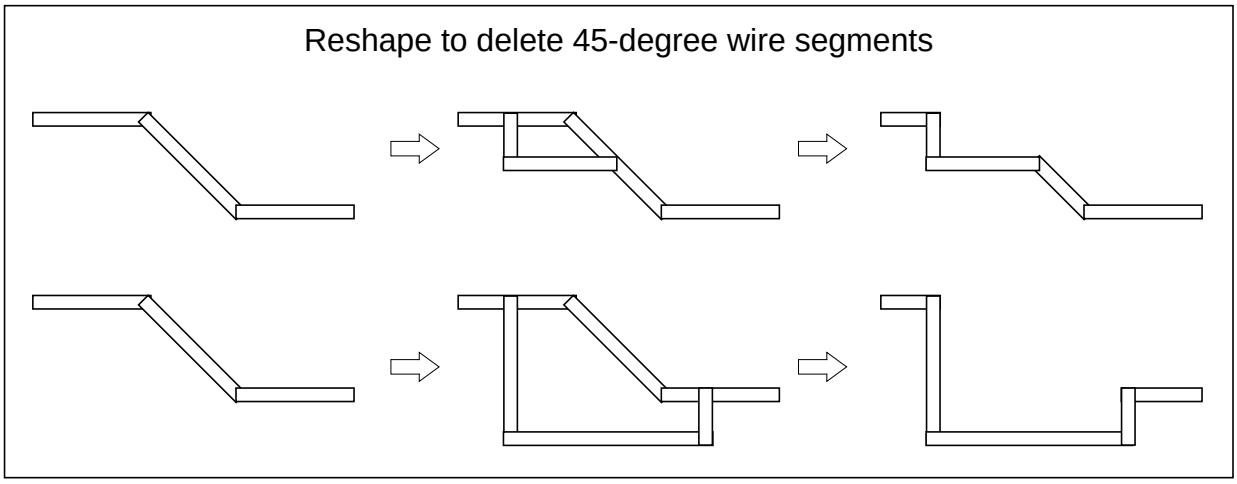
editAddRoute

```
editAddRoute x y
     [-reshape]
```

Creates a wire segment that starts or ends at the specified coordinates. Splits diagonal wires whose width is greater than the LEF `MAXWIDTH` parameter for that layer as they are drawn. This command also supports the reshaping of 45-degree wire segments.

Parameters

x y	Specify the x and y coordinates for the startpoint or endpoint of the wire segment.
`-reshape`	Reshapes an existing 45-degree wire segment.



Example

- The following commands create wire segments that start and stop at the specified points. The wire ends at the point specified by `editCommitRoute`.

```
editAddRoute 33.392 36.831
editAddRoute 32.985 32.766
editAddRoute 18.064 32.115
editAddRoute 20.951 27.073
editCommitRoute 20.992 27.114
```

editAddVia

`editAddVia` *x y*

Creates a via instance and places the origin at the specified coordinates.The origin of the instance's via cell may or may not be at its center. Specify the via parameters with the setViaEdit command.

Parameters

x y	Specify the x and y coordinates for the via.

editChangeLayer

```
editChangeLayer
     -layer_horizontal layer
     -layer_vertical layer
```

Changes the layer of selected wires and updates vias connecting to the wires that change layers.

Note: Use the editSelect command to select the wires.

Parameters

`-layer_horizontal` *`layer`*

Specifies the new layer for the selected horizontal wires.

`-layer_vertical` *`layer`*

Specifies the new layer for the selected vertical wires.

Example

- The following command moves all selected horizontal wires to the *M3* layer and all selected vertical wires to the *M4* layer:

  ```
  editChangeLayer -layer_horizontal M3 -layer_vertical M4
  ```

editChangeNet

`editChangeNet -to` *`netName`*

Changes the net associated with selected wires and vias.

Parameters

`-to`	Specifies the name of the net to which to assign the selected wires and vias.

Example

- The following commands select wires and vias that belong to net `VDD1` and reassign them to net `VDD2`:

```
editSelect -nets VDD1
editSelectVia -nets VDD1
editChangeNet -to VDD2
```

editChangeRule

```
editChangeRule
     -net net_name
     -from old_rule
     -to new_rule
```

Changes the LEF rule associated with wire segments on the specified net from the *old_rule* to the *new_rule*.

Note: This command does not change the routing rule for the net. If the net is re-routed by NanoRoute, these changes will be lost. To change the routing rule, use `setAttribute -net -non_default_rule`. For more information, see "setAttribute" on page 1271.

Parameters

`-from` *old_rule*

Specifies the qualifying rule. Only wire segments whose associated rule matches *old_rule* are changed.

`-net` *net_name*

Specifies the net to which to apply the changes when you run this command.

`-to` *new_rule*

Specifies the destination rule. Wire segments originally associated with *old_rule* are associated with *new_rule* after this command runs.

Specify one of the following:

- A nondefault rule defined in the LEF LAYER (Routing) section
- `default` (applies the default routing rule)

Example

- The following command changes the wire segments in net XYZ that are associated with rule SP to the default rule. If the wire width or extension of SP is different from its counterpart in the default rule, the wire geometry is changed accordingly.

  ```
  editChangeRule -net XYZ -from SP -to default
  ```

editChangeStatus

```
editChangeStatus -to {COVER | FIXED | ROUTED | SHIELD netName | NOSHIELD}
```

Changes the DEF status of selected wires or vias to the specified status.

Note: Use the `editSelect` command to select the wires.

Parameters

`-to {COVER | FIXED | ROUTED | SHIELD netName| NOSHIELD}`

Specifies the DEF status of the selected wires or vias after you issue this command. Type the status in all upper-case letters.

Note: You cannot change a special wire to `NOSHIELD` status or change a signal wire to `SHIELD` status.

editChangeVia

```
editChangeVia
     -from {old_via_name | old_via_list}
     -to {new_via_name | new_via_list}
     {-area {x1 y1 x2 y2} | -at {x y}}
     [-signal_only {0 | 1}]
     [-net net_name]
```

Changes the vias touching the specified point or area to use *new_via_name* as their new via master. Does not change vias whose masters are not *old_via_name*.

Parameters

-area {*x1 y1 x2 y2*}

Specifies an area in which to replace vias. If a via indicated by the -from parameter is in this area, replaces the via with the via indicated by the -to parameter.

Note: You must specify the -area parameter or the -at parameter. If you do not specify either parameter, the software does not change any vias.

-at {*x y*}

Specifies the coordinates for the origin of a via to replace. The via at this location is replaced by the via indicated by the -to parameter.

Note: You must specify either the -area parameter or the -at parameter. If you do not specify either parameter, the software does not change any vias.

-from {*old_via_name | old_via_list*}

Specifies a via or list of vias to replace. Enclose the via names on the list in curly braces and separate each name with a space.

Note: The number of via names on the list must be equal to the number of via names on the list for the -to parameter. In addition, the corresponding pair of vias in each list must be on the same layer.

-net *net_name*

Specifies the net name on which to change vias.

`-signal_only {1 | 0}` Specifies whether signal vias only or signal and special vias are changed. Specify `1` to change signal via only. Specify `0` to change signal and special vias.
Default: `0`

`-to {`*`old_via_name | old_via_list`*`}`

Specifies a via or a list of vias to replace the vias indicated by the `-from` parameter. If you specify more than one via, enclose the via list in curly braces and separate each via name with a space.

Note: The number of via names on the list must be the equal to the number of via names on the list for the `-from` parameter. In addition, the corresponding pair of vias in each list must be on the same layer.

editChangeWidth

```
editChangeWidth
     -width_horizontal width
     -width_vertical width
```

Changes the width of special wires and updates vias connected to the changed wires to maintain the via-to-wire dimension ratio. If you change the width of a wire, and then change it back to the original width, you might see a discrepancy due to rounding.

This command applies to special wires only. The width of a signal wire always comes from the LEF rule used to create it.

Note: Use the editSelect command to select the wires.

Parameters

`-width_horizontal` *`width`*

Specifies, in microns, the new width for selected horizontal special wires.

`-width_vertical` *`width`*

Specifies, in microns, the new width for selected vertical special wires.

Example

- The following command changes the width of all selected horizontal special wires to 0.25 μm and all selected special vertical wires to 0.45 μm:

  ```
  editChangeWidth -width_horizontal 0.25 -width_vertical 0.45
  ```

editCommitPoly

`editCommitPoly` *x y*

Ends a newly created polygon at the specified coordinates. Use this command after at least one `editAddPoly` command has been issued.

Parameters

x y	Specify the x and y coordinates for the endpoint of a newly created wire polygon.

editCommitRoute

`editCommitRoute` *x y*

Ends a newly created wire segment at the specified coordinates. Splits diagonal wires whose width is greater than the LEF `MAXWIDTH` parameter for that layer as they are drawn. Use this command after at least one `editAddRoute` command has been issued.

Parameters

x y	Specify the x and y coordinates for the endpoint of a newly created wire segment.

Example

- The following commands create a wire that ends at the point specified by `editCommitRoute`:

```
editAddRoute 33.392 36.831
editAddRoute 32.985 32.766
editAddRoute 18.064 32.115
editAddRoute 20.951 27.073
editCommitRoute 20.992 27.114
```

editCutWire

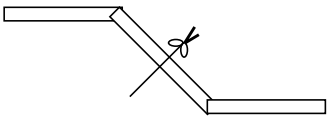
`editCutWire -x1 x1 -y1 y1 -x2 x2 -y2 y2`

Cuts wires or rectangles at the specified location. The cut line must go across the wire or rectangle from one edge to the other.

Once cut, signal wire pieces maintain a one-half wire-width extension. Special wires do not maintain an extension.

Note: This command also supports 45-degree wire edits.

Cut wire to support 45-degree wire segments



Parameters

`-x1 x1 -y1 y1 -x2 x2 -y2 y2`

Specify the coordinates of the line segment that cuts the wires or rectangles.

Example

- The following command cuts wires at the specified coordinates:

```
editCutWire -x1 200 -y1 300 -x2 200 -y2 310
```

editDelete

```
editDelete
     [-area lx ly hx hy]
     [-direction {H | V}]
     [-layers {layer | {list_of_layers}}]
     [-nets {net | {list_of_nets}]
     [-shield {0 | 1}]
     [-objects {Selected | All}]
     [-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE]
         [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE] [FILLWIREOPC] [DRCFILL]}
         | None}]
     [-status {[COVER] [FIXED] [NOSHIELD] [ROUTED] [SHIELD]}
     [-type {Signal | Special}]
     [-use CLOCK]
     [-wires_only {0 | 1}]
```

Deletes wires. Also deletes vias and violation markers associated with the wires.

The `editDelete` command deletes physical wires. The `deleteNet` command deletes logical nets. For more information, see `deleteNet` on page 921.

Parameters

`-area lx ly hx hy` Specifies the coordinates for the area from which to delete the wires.
Default: Deletes all wires.

lx	Specifies the lower left x coordinate.
ly	Specifies the lower left y coordinate.
hx	Specifies the upper right x coordinate.
hy	Specifies the upper right y coordinate.

`-direction {H | V}` Specifies the orientation of the wires to delete.
Default: If you do not specify this parameter, deletes wires with either orientations.

H	Specifies that only horizontal wires are to be deleted.
V	Specifies that only vertical wires are to be deleted.

`-layers {`*`layer`* `|` **`{`***`list_of_layers`***`}`**`}`

Specifies the layers from which to delete wires. You must specify layers in the following format: `M`*`number`*, for example, `M2`. If you do not use this format, `editDelete` does not delete any wires and does not generate an error message. If you specify multiple layers, enclose the list of layers in curly braces.
Default: If you do not specify this parameter, deletes wires from all layers.

`-nets {net |` **`{`***`list_of_nets`***`}`**`}`

Specifies the net or list of nets from which to delete wires.

Note: You can use wildcard values in this field. For example, if you specify `V*`, deletes wires from nets `VDD` and `VSS`.

`-objects {Selected | All}`

Specifies whether to delete all wires or selected wires. Select the wires in the design display area interactively or use the `editSelect` command.
Default: If you do not specify this parameter, deletes all wires.

`-shapes {`**`{`**`[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE] [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE] [FILLWIREOPC] [DRCFILL]`**`}`**`} | None}`

Specifies the shapes of the wires to delete. The status must be in all upper-case letters. If you specify multiple shapes, enclose the shapes in braces. For shape definitions, see Shapes, in the "DEF Syntax" chapter of the *LEF/DEF Language Reference*.
Default: If you do not specify this parameter, deletes wires of all shapes.

`-shield {0 | 1}`

Deletes the wires and vias shielding the nets specified by the `-nets` parameter.

`-status {`**`{`**`[COVER] [FIXED] [NOSHIELD] [ROUTED] [SHIELD]`**`}`**`}`

Specifies the status of wires to delete. The status must be in all upper-case letters. If you specify more than one status, enclose the status list in curly braces.

Note: If you select `-type Special -status NOSHIELD` or `-type Signal -status SHIELD` the software generates an error message.

`-type {Signal | Special}`

Specifies whether to delete signal wires or special wires.

Note: If you specify `Signal`, omit the `-shapes` parameter or specify `-shapes None`.

`-use CLOCK` Specifies that only `CLOCK` wires will be deleted.

`-wires_only {0 | 1}` Deletes all the special wires in the net, leaving the vias untouched.

Examples

- The following command removes wires from the *M1* and *M2* layers:

  ```
  editDelete -layers M1 M2
  ```

- The following command removes horizontal fillwires from the *M1* layer:

  ```
  editDelete -layers M1 -direction H -shapes FILLWIRE
  ```

- The following command removes wires from the specified area:

  ```
  editDelete -area 100 100 200 200 -objects All
  ```

- The following command removes the selected wires from the specified area:

  ```
  editDelete -area 100 100 200 200 -objects Selected
  ```

- The following command only deletes the special wires that belong to the net carry4 leaving the vias untouched:

  ```
  editDelete -type Special -nets carry4 -wires_only 1
  ```

editDeleteFillet

```
editDeleteFillet
```

Deletes the tear-drop fillet that was added between a wire and bump connection with the editAddFillet command.

editDeleteViolations

```
editDeleteViolations
     -keep_fixed
```

Deletes signal nets with violation markers.

Parameters

`-keep_fixed`	Does not delete wires marked `FIXED` in the DEF file when deleting nets with violations.

editDeselect

```
editDeselect
     [-area x1 y1 x2 y2]
     [-direction {H | V}]
     [-layers {layer | {list_of_layers}}]
     [-nets {net | {list_of_nets} [-shield {0 | 1}]}]
     [-objects {Selected | All}]
     [-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE]
          [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE] [FILLWIREOPC] [DRCFILL]}}
     | None}
     [-status {{[COVER] [FIXED] [NOSHIELD] [ROUTED] [SHIELD]}}]
     [-type {Signal | Special}]
     [-use CLOCK]
```

Deselects wires. Use this command when nets are selected.

Tip

You can use this command to deselect nets by property. For example, to delete all signal nets except clock nets, select all signal nets with editSelect, deselect the clock nets with this command, then delete the selected nets with editDelete.

Parameters

`-area` *x1 y1 x2 y2*

Specifies the coordinates for the area from which to deselect wires.
Default: Selects all wires.

x1	Specifies the lower left x coordinate.
y1	Specifies the lower left y coordinate.
x2	Specifies the upper right x coordinate.
y2	Specifies the upper right y coordinate.

`-direction {H | V}`

Specifies the orientation of the wires to deselect.
Default: If you do not specify this parameter, selects wires in both orientations.

H	Specifies horizontal wires.
V	Specifies vertical wires.

`-layers {`*`layer`* `| {`*`list_of_layers`*`}}`

Specifies the layers from which to deselect wires. If you specify multiple layers, enclose the layer names in curly braces.
Default: If you do not specify this parameter, deselects wires from all layers.

`-nets {`*`net`* `| {`*`list_of_nets`*`}}`

Specifies the net names from which to deselect wires.

Note: You can use wildcard values in this field. For example, if you specify `V*`, deselects wires from nets `VDD` and `VSS`.

`-objects {Selected | All}`

Specifies whether to deselect all wires or selected wires.

Default: If you do not specify this parameter, deselects all wires.

`-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE] [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE] [FILLWIREOPC] [DRCFILL]} | None}`

Specifies the shapes of the wires to deselect. Use all upper-case letters for the shapes. If you specify multiple shapes, enclose the list of shapes in braces. For shape definitions, see Shapes, in the “DEF Syntax” chapter of the *LEF/DEF Language Reference*.
Default: If you do not specify this parameter, deselects wires of all shapes.

`-shield {0 | 1}`

Deselects the shield nets associated with the nets specified by the `-nets` parameter.

`-status {{[COVER] [FIXED] [NOSHIELD] [ROUTED] [SHIELD]}}`

Specifies the status of the wires to deselect. Use all upper-case letters for the status. If you specify more than one status, enclose the status list in curly braces.

Note: If you select `-type Special -status NOSHIELD` or `-type Signal -status SHIELD` the software generates an error message.

`-type {Signal | Special}`

Specifies whether to deselect signal wires or special wires.

Note: If you specify `Signal`, you must omit the `-shapes` parameter or specify `-shapes None`.

`-use CLOCK` Specifies that only `CLOCK` wires will be deselected.

Example

- The following command deselects all horizontal fillwires on the *M1* layer:

```
editDeselect -layers M1 -direction H -shapes FILLWIRE
```

editDeselectVia

```
editDeselectVia
     [-area x1 y1 x2 y2]
     [-cut_layers {cut_layer | {list_of_cut_layers}}]
     [-nets {net | {list_of_nets}}]
     [-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE]
         [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE]} | None}
     [-status {[COVER] [FIXED] [NOSHIELD] [ROUTED] [SHIELD]}]
     [-type {Special | Signal}]
     [-use CLOCK]
```

Deselects vias using the specified parameters.

Parameters

`-area x1 y1 x2 y2`

Specifies the coordinates for the area from which to deselect vias.
Default: Deselects all vias.

`-cut_layers {cut_layer | {list_of_cut_layers}}`

Specifies the cut layers from which to deselect vias. If you specify multiple cut layers, enclose the layer names in curly braces.
Default: If you do not specify this parameter, deselects vias from all cut layers.

`-nets {net | {list_of_nets}}`

Deselects vias on the specified nets. You can use wildcard values in this field. For example, if you specify `V*`, deselects vias from nets `VDD` and `VSS`.

`-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE] [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE]} | None}`

Specifies the shapes of the vias to deselect. Use all upper-case letters for the shapes. If you specify multiple shapes, enclose the list of shapes in curly braces.
Default: If you do not specify this parameter, deselects vias of all shapes.

`-status {[COVER] [FIXED] [NOSHIELD] [ROUTED] [SHIELD]}`

Specifies the status of the deselected vias. The status must be in all upper-case letters. If you specify more than one status, enclose the status list in curly braces.

Note: If you select `-type Special -status NOSHIELD` or `-type Signal -status SHIELD` the software generates an error message.

`-type {Special | Signal}`

Specifies whether the deselected vias are signal net or special net vias.

Note: If you specify `Signal`, omit the `-shapes` parameter or specify `-shapes None`.

`-use CLOCK` Specifies that only `CLOCK` vias will be deselected.

editDuplicate

```
editDuplicate
     -layer_horizontal layer
     -layer_vertical layer
```

Copies the geometry of selected special wires and pastes them at the same coordinates. If you do not specify a layer, the software pastes them onto the same layer. If you specify a layer, the software copies them to the specified layer and generates vias to maintain connectivity for the duplicated wire.

After duplicating the wires, the software selects the new wires and deselects the original wires. Use the editMove command to move duplicated wires to new locations.

Note: Use the `editSelect` command to select wires.

Parameters

`-layer_horizontal` *`layer`*

Specifies the new layer for duplicated horizontal wires.

`-layer_vertical` *`layer`*

Specifies the new layer for duplicated vertical wires.

Example

- The following command duplicates selected horizontal wires and places them on the *M3* layer and duplicates selected vertical wires and places them on the *M4* layer. It generates vias for the new wires.

  ```
  editDuplicate -layer_horizontal M3 -layer_vertical M4
  ```

editFixWideWires

`editFixWideWires`

Splits wires that violate the `MAXWIDTH` value in the LEF `LAYER` (Routing) statement. Also splits diagonal wires whose width is greater than the `MAXWIDTH` value as they are drawn.

Parameters

None

editMerge

`editMerge`

Merges all wires of the same width, layer, and net that are colinear with and electrically connected to a selected wire. Does not merge signal wires that contain branches. This command also supports 45-degree wire edits.

Note: Use the `editSelect` command to select the wire.

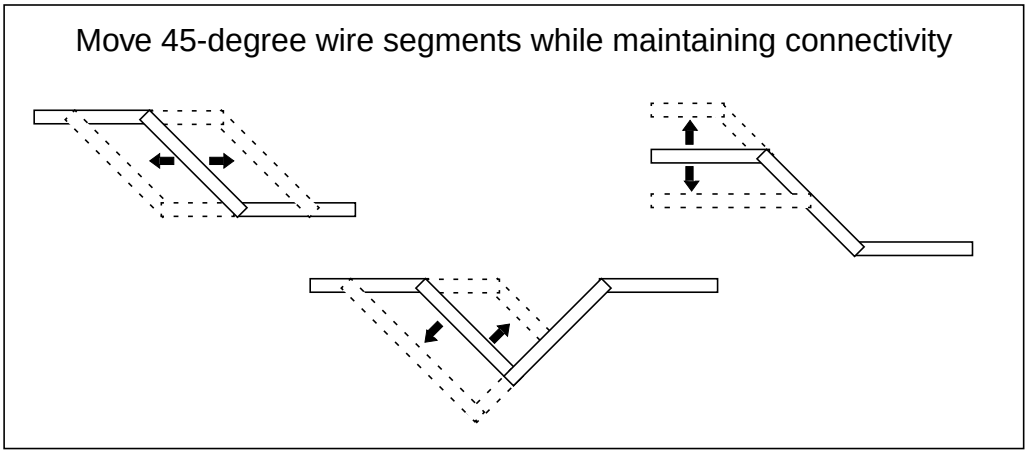
Parameters

None

editMove

```
editMove
     {x | y | diag45 | diag135}
     distance
```

Moves selected wires in the orthogonal direction. Maintains connectivity by stretching wires connected in the orthogonal direction. This command also supports 45 and 135-degree wire edits.



Parameters

`diag45`	Moves selected wires the specified *`distance`* in a 45-degree direction.
`diag135`	Moves selected wires the specified *`distance`* in a 135-degree direction.
`distance`	Specifies, in microns, the distance to move the wire. *Tip* If the `-snap_to_track` parameter in the `setEdit` command is set to `1`, the wire is snapped to the track closest to the specified *`distance`*.
x	Moves selected vertical wires to the right (if you specify a positive value for *`distance`*) or left (if you specify a negative value for *`distance`*).

y	Moves selected horizontal wires up (if you specify a positive value for *`distance`*) or down (if you specify a negative value for *`distance`*).

editSelect

```
editSelect
     [-area x1 y1 x2 y2]
     [-direction {H | V}]
     [-layers {layer | {list_of_layers}}]
     [-nets {netName | {list_of_netNames} [-shield {0 | 1}]}
     [-objects {Selected | All}]
     [-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE]
         [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE] [FILLWIREOPC] [DRCFILL]}} |
     None}
     [-status {[COVER] [FIXED] [NOSHIELD] [ROUTED] [SHIELD] }]
     [-type {Signal | Special}]
     [-use CLOCK]
```

Selects wires. The wires are updated by subsequent wire editing commands.

Parameters

`-area` *x1 y1 x2 y2*
: Specifies the coordinates for the area from which to select wires.

  Default: Entire area

  x1 — Specifies the lower left x coordinate.

  y1 — Specifies the lower left y coordinate.

  x2 — Specifies the upper right x coordinate.

  y2 — Specifies the upper right y coordinate.

`-direction {H | V}`
: Specifies the orientation of the wires to select.
  Default: If you do not specify this parameter, selects wires with either orientation.

  `H` — Specifies horizontal wires.

  `V` — Specifies vertical wires.

`-layers {`*layer* | **{***list_of_layers***}**`}`
: Specifies the layers from which to select wires. If you specify multiple layers, enclose the layer names in curly braces.
  Default: If you do not specify this parameter, selects wires on all layers.

`-nets {`*`net`* `| {`*`list_of_nets`*`}}`

Specifies the net names from which to select wires.

You can use wildcard values in this field. For example, if you specify `V*`, selects wires from nets `VDD` and `VSS`.

`-objects {Selected | All}`

Specifies whether to select all wires.

Default: If you do not specify this parameter, selects all wires.

`-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE] [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE] [FILLWIREOPC] [DRCFILL]} | None}`

Specifies the shapes of the wires to select. Use all upper-case letters for the shapes. If you specify multiple shapes, enclose the list of shapes in braces. For shape definitions, see Shapes, in the "DEF Syntax" chapter of the *LEF/DEF Language Reference*.
Default: If you do not specify this parameter, selects all shapes.

`-shield {0 | 1}`

When set to `1`, selects the shield wires associated with the nets specified by the `-nets` parameter.

`-status {{[COVER] [FIXED] [NOSHIELD][ROUTED] [SHIELD]}}`

Specifies the status of wires to select. Use all upper-case letters for the status. If you specify more than one status, enclose the status list in curly braces.

Note: If you select `-type Special -status NOSHIELD` or `-type Signal -status SHIELD` the software generates an error message.

`-type {Signal | Special}`

Specifies whether to select signal wires or special wires.

Note: If you specify `Signal`, you must omit the `-shapes` parameter or specify `-shapes None`.

`-use CLOCK` Specifies that only `CLOCK` wires will be selected.

Example

- The following command selects all horizontal fillwires on the *M1* layer:

```
editSelect -layers M1 -direction H -shapes FILLWIRE
```

editSelectVia

```
editSelectVia
     [-area x1 y1 x2 y2]
     [-cut_layers {cut_layer | {list_of_cut_layers}}]
     [-nets {net | {list_of_nets}}]
     [-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE]
         [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE]} | None}
     [-status {[COVER] [FIXED] [NOSHIELD [ROUTED] [SHIELD]}]
     [-type {Special | Signal}]
     [-use CLOCK]
```

Selects vias using the specified parameters.

Parameters

`-area x1 y1 x2 y2`

Specifies the coordinates for the area from which to select vias.
Default: Selects all vias.

`-cut_layers {cut_layer | {list_of_cut_layers}}`

Specifies the cut layers from which to select vias. If you specify multiple cut layers, enclose the layer names in curly braces.
Default: If you do not specify this parameter, selects vias from all cut layers.

`-nets {net | {list_of_nets}}`

Selects vias on the specified nets. You can use wildcard values in this field. For example, if you specify `V*`, selects vias from nets `VDD` and `VSS`.

`-shapes {{[RING] [STRIPE] [FOLLOWPIN] [IOWIRE] [COREWIRE] [BLOCKWIRE] [PADRING] [BLOCKRING] [FILLWIRE]} | None}`

Specifies the shapes of the vias to select. Use all upper-case letters for the shapes. If you specify multiple shapes, enclose the list of shapes in curly braces.
Default: If you do not specify this parameter, selects vias of all shapes.

`-status {[COVER] [FIXED] [NOSHIELD] [ROUTED] [SHIELD]}`

Specifies the status of the selected vias. The status must be in all upper-case letters. If you specify more than one status, enclose the status list in curly braces.

Note: If you select `-type Special -status NOSHIELD` or `-type Signal -status SHIELD` the software generates an error message.

`-type {Special | Signal}`

Specifies whether the selected vias are signal net or special net vias.

Note: If you specify `Signal`, omit the `-shapes` parameter or specify `-shapes None`.

`-use CLOCK` Specifies that only `CLOCK` vias will be selected.

editSplit

`editSplit`

Splits selected special wires at connection points with orthogonal wires.

Note: Use the `editSelect` command to select the wires.

Parameters

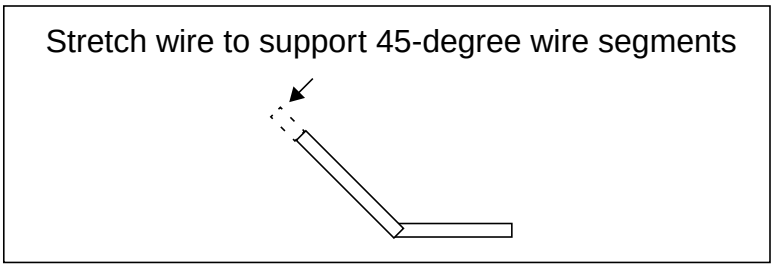
None

editStretch

```
editStretch {high | low}
     distance
     {x | y | diag45 | diag135}
```

Stretches or shrinks selected wires and snaps them to the closest manufacturing grid point. This command also supports 45 and 135-degree wire edits.

Note: Use the `editSelect` command to select the wires.



Parameters

`diag45`	Stretches selected wires the specified *`distance`* in a 45-degree direction.
`diag135`	Stretches selected wires the specified *`distance`* in a 135-degree direction.
`distance`	Specifies the distance in microns to stretch or shrink wires. Use positive number to stretch wires and a negative number to shrink wires.
`high`	Stretches or shrinks wires from the top or the right.
`low`	Stretches or shrinks the wire from the bottom or the left.
`x`	Stretches or shrinks wires horizontally.
`y`	Stretches or shrinks wires vertically.

Examples

- The following command stretches the top of the selected vertical wire by 18.5 µm.

  ```
  editStretch y 18.5 high
  ```

- The following command shrinks a horizontal wire from the left by 12.8 µm.

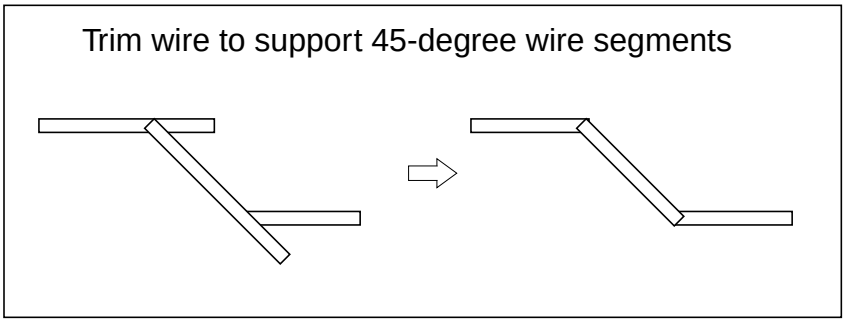
  ```
  editStretch x -12.8 low
  ```

editTrim

```
editTrim
     [-selected | -all | -nets netNames]
     [-type float]
```

Removes or trims the following wires:

- Trims floating stubs of wires back to the closest connection point.
- Removes wires that do not have electrical connections to other wires or pins (floating wires).
- Removes wires with only one connection to another wire or pin, along with connecting vias. Removing a wire with one connection might cause the wire to which the just-removed wire is connected to change from being a wire with two connections to a wire with one connection and thus it will also be removed. This process is repeated until no wires with one connection are left.
- Trims wires to support 45-degree wire segments.



Note: This command does not trim metal fill segments. To trim metal fill segments use the trimMetalFill command.

Parameters

`-all`	Trims all special wires and floating stubs.
`-nets` *`netNames`*	Trims wires on the nets you specify. You can use wildcards to specify the nets.
`-selected`	Trims selected wires. Use the `editSelect` command to select the wires. This is the default value.

`-type float`	Removes wires that do not have electrical connections to other wires or pins (floating wires).

setEdit

```
setEdit
     [-allow_all_rules {0 | 1}]
     [-arrow_increment value]
     [-at_a_turn {Reverse Order | Keep Order}]
     [-close_polygons {0 | 1}]
     [-create_crossover_vias {0 | 1}]
     [-create_via_on_pin {0 | 1}]
     [-drawing_wire value]
     [-extend_end_wires {0 | 1}]
     [-extend_start_wires {0 | 1}]
     [-extend_to_bdry {0 | 1}]
     [-force_regular {0 | 1}]
     [-force_special {0 | 1}]
     [-layer_horizontal name]
     [-layer_vertical name]
     [-look_down_layers integer]
     [-look_up_layers integer]
     [-move_wires_with_block {0 | 1}]
     [-nets {net | {list_of_nets}}]
     [-nets_shielding {netNames}]
     [-outer_shield_spacing value]
     [-outer_shield_width value]
     [-override {0 | 1}]
     [-override_spec {specification}]
     [-reshape {0 | 1}]
     [-rule rule]
     [-shape {RING | STRIPE | FOLLOWPIN | IOWIRE | COREWIRE | BLOCKWIRE
         | PADRING | BLOCKRING | FILLWIRE | DRCFILL | None}]
     [-shield_adjacent_wires {0 | 1}]
     [-shield_high {0 | 1}]
     [-shield_look_up_layers integer]
     [-shield_look_down_layers integer]
     [-shield_low {0 | 1}]
     [-snap_align_to alignment]
     [-snap_to_pg_wire {0 | 1}]
     [-snap_to_pin {0 | 1}]
     [-snap_to_row {0 | 1}]
     [-snap_to_track {0 | 1}]
     [-snap_to_track_regular {0 | 1}]
     [-spacing_horizontal value]
     [-spacing_vertical value]
     [-split_wide_wires {0 | 1}]
     [-stop_at_drc {0 | 1}]
     [-stretch_end {high | low}]
     [-use_wire_group {0 | 1}]
     [-use_interleaving_wire_group {0 | 1}]
     [-use_wire_group bits number_of_bits]
     [-use_wire_group_enforcement {0 | 1}]
     [-use_wire_group_enforcement_spacing spacing]
```

```
    [-use_wire_group_reinforcement_width width]
    [-use_wire_group_enforcement_group_via {0 | 1}]
    [-width_horizontal value]
    [-width_vertical value]
```

Updates the Edit Route form and the design display area.

Parameters

`-allow_all_rules {0 | 1}`

If a net has a non-default rule attached to it, its wires can only be created with that non-default rule or the default rule. Set this parameter to 1 to allow any nondefault rule to be used, regardless of whether a net has a nondefault rule.

`-arrow_increment` *`value`*

Specifies the step increments in microns to move wires with the arrow keys.
Default: `1.0`.

`-at_a_turn {Reverse Order | Keep Order}`

Specifies whether the order of the wires changes or stays the same when the wire makes a 90-degree turn.
Default: `Keep Order`

`-close_polygons {0 | 1}`

Specifies whether to close a special route structure toward itself, using the `Escape` key. For the closing to complete, the ending wire segments must be drawn towards the start wire segments, but do not have to touch them. If you specify this parameter, the software ignores the `-extend_start_wires` parameter and the `-extend_end_wires` parameter.
Default: `0`

`-create_crossover_vias {0 | 1}`

Specifies whether the software creates a via when you draw a wire that crosses a wire or pin of the same net that is on a different layer. Does not create vias if minimum or maximum layer constraints prevent the creation of vias. If you draw a wire between two pins, the software creates vias on the pins as needed, regardless of whether this option is selected or deselected.

If you specify `0` for this parameter, the software does not create vias at crossovers. For example, if you draw a wire from left to right, the software begins creating a via at the left edge of the vertical target. This via is removed as soon as the drawing wire moves past the right edge of the vertical target.
Default: `1`

`-create_via_on_pin {0 | 1}`

Specifies whether the software creates vias at pins. Specify `0` to prevent the software from creating vias at pins. A value of `0` might be needed for bump pins that are not square-shaped.
Default: `1`

`-drawing_wire` *`value`*

Specifies which of the nets used with the `-nets` parameter corresponds to the mouse pointer location when adding an array of wires. For horizontal wires, a value of `1` indicates the bottom-most net. For vertical wires, a value of `1` indicates the left-most net. The *`value`* is an integer of the same or lower value as the number of nets. For example, if you specify 5 nets, you can specify `1`, `2`, `3`, `4`, or `5` as the value for this parameter.

This net is also used as the reference net when changing direction. All other nets around this net are lengthened or shortened accordingly.
Default: `1`

`-extend_end_wires {0 | 1}`

Specifies whether, after completing a signal or power route, the endpoint of the route segment extends and connects to the first logical target. If there is no logical target, the endpoint is not changed.
Default: `0`

`-extend_start_wires {0 | 1}`

Specifies whether, after completing a signal or power route, the startpoint of the route segment extends and connects to the first logical target. If there is no logical target, the startpoint is not changed.
Default: `0`

`-extend_to_bdry {0 | 1}`

Specifies whether, after completing a special route, the startpoint or endpoint of the segment always extends to the cell boundary. If the segment is extended to the boundary and crosses wires on the same net, makes a via connection to the wires. In addition, creates pins at the boundary. This parameter can only be used in conjunction with the `-extend_start_wires` parameter, the `-extend_end_wires` parameter, or both.
Default: `0`

`-force_regular {0 | 1}`

Creates regular wires and vias, even if special (power or ground) nets are specified.
Default: `0`

`-force_special {0 | 1}`

Specifies whether to create special wires for signal nets.
Default: `0`

`-layer_vertical` *`name`*

Specifies the layer for vertical wires.

`-look_up_layers` *`integer`*

Specifies the number of layers above the current layer that added wires will connect to with a via.
Default: `100` (all layers)

`-look_down_layers` *`integer`*

Specifies the number of layers below the current layer that added wires will connect to with a via.
Default: `100` (all layers)

`-move_wires_with_blocks {0 | 1}`

Specifies whether to move wires that are connected to a block when the block moves.
Default: `0`

`-nets {`*`net`* `| {`*`list_of_nets`*`}}`

Specifies one or more nets for editing. If you specify multiple nets, separate each net name with a space and enclose the names in curly braces.

You can use the following format to specify a bus:

`netName` `[`*`x`*`:`*`y`*`]`

For example, to draw eight wires for a bus, type the following:

`-nets` *`netName`*`[7:0]`

This is equivalent to specifying *`netName`*`[7], `*`netName`*`[6], ... `*`netName`*`[0]`. Specifying `-nets` *`netName`*`[0:7]` reverses the order.

The order in which you specify the nets determines the drawing arrangement of the wires.

- For horizontally drawn wires, the first net specified is drawn as the bottom-most wire, followed by the next net on its top side, and so on. The last net specified is drawn as the top-most wire.
- For vertically drawn wires, the first net specified is drawn as the left-most wire, followed by the next net, and so on. The last net specified is drawn as the right-most wire.

`-nets_shielding {`*`netNames`*`}`

Specifies net names for shield wires, usually power or ground names.

`-outer_shield_spacing` *`value`*

Specifies a spacing value in microns to be used by the `-shield_low` and `-shield_high` parameters.
Default: `0` (minimum spacing on the layer)

`-outer_shield_width` *`value`*

Specifies a width value in microns to be used for outer shield(s) depending on the values of `-shield_low` or `-shield_high` parameters.
Default: `0` (minimum wire width on the layer)

Note: You must define the outer shield first. The outer shield is the shielding wire beyond the first or the last wire being created manually.

`-override {0 | 1}`

Specifies whether to use the override specification.
Default: `0`

`-override_spec {`*`specification`*`}`

Contains the details of the override specification.

Specify the net numbers, followed by the override width and the override spacing to the next net.

The net number is the position of the net specified by the `-nets` parameter. If you specify more than one net, separate net numbers with commas.

For example, the following specification indicates that the first and second nets should have a width of 10.0 and a spacing of 3.0:

```
1,2 10.0 3.0
```

`-reshape {0 | 1}`

Specifies that when you add new wires, existing redundant wires within the route are automatically removed.

`-rule` *`rule`* Uses the specified LEF rule for regular wires.

`-shape {RING | STRIPE | FOLLOWPIN | IOWIRE | COREWIRE | BLOCKWIRE | PADRING | BLOCKRING | FILLWIRE | DRCFILL | None}`

Specifies the shape associated with the wire you draw. Use all upper-case letters for the shape.
Default: `STRIPE`

`-shield_adjacent_wires {0 | 1}`

Specifies whether to add a minimum width shield wire(s) centered between two adjacent wires.
Default: `0`

`-shield_high {0 | 1}`

Specifies whether to add minimum width shield wires at the top (High) side of horizontal routes or at the right side of vertical routes. The spacing of the added shield wire is determined by the value of the `-outer_shield_spacing` parameter. If this value is 0, the spacing is determined by the minimum spacing rule in the technology file.
Default: `0`

`-shield_look_down_layers` *`integer`*

Specifies the number of layers below the current layer that an added shield wire can connect using a via.

`-shield_look_up_layers` *`integer`*

Specifies the number of layers above the current layer that an added shield wire can connect using a via.

`-shield_low {0 | 1}`

Specifies whether to add minimum width shield wires at the bottom (Low) side of horizontal routes or at the left side of vertical routes. The spacing of the added shield wire is determined by the value of the `-outer_shield_spacing` parameter. If this value is 0, the spacing is determined by the minimum spacing rule in the technology file.
Default: `0`

`-snap_align_to {Center | Low | High | Auto}`

Specifies how to align with a pin when ending a route. To ensure the proper alignment, set `-snap_to_track` to `0`.
Default: `Auto`

Select one of the following:

`Center`	Snaps wires to the center of the pin.
`Low`	Snaps wires to the bottom of vertical pins or to the left side of horizontal pins.
`High`	Snaps wires to the top of vertical pins or to the right side of horizontal pins.
`Auto`	The software determines alignment automatically.

`-snap_to_pg_wire {0 | 1}`

Snaps new wires whose start point is within a “snap region” to the closest end of a power or ground wire on the same net. The snap region is one-half the width of the target wire.

- The snap region for orthogonal wires forms a square with the width of the power or ground wire.
- The snap region for wires at 45-degree angles to the power or ground wire forms a square from the point where the wire will connect. Each side of the square is the width of the power or ground wire.
- The snap region for wires that are parallel to power or ground wires is the full power or ground wire.

Default: `1`

Note: When you specify `1` for both `-snap_to_pg_wire` and `-snap_to_track`, `-snap_to_pg_wire` takes precedence.

`-snap_to_pin {0 | 1}`

Specifies whether to snap added or moved wires to a pin automatically. If you double-click at a pin, it snaps the wire to the pin. To ensure the wire is snapped to the center of the pin, set `-snap_to_track` (or `-snap_to_track_regular`) to 0 and specify a value for `-snap_align_to`.

When you specify `1` for both `-snap_to_track` (or `-snap_to_track_regular`) and `-snap_to_pin`, the software has the following behavior:

- If the wire will fall partially outside the pin when snapped to a track, it snaps to the pin.
- If the wire will fall inside the pin when snapped to a track, it snaps to both the pin and a track.
- If there is no track on the pin, the wire snaps to the pin.

Default: `1`

`-snap_to_row {0 | 1}`

Snaps added or moved wires to the nearest standard cell row. Use this parameter to create followpins. This parameter supports both horizontal and vertical rows.

Note: `-snap_to_track` has priority over this parameter if both are specified.

`-snap_to_track {0 | 1}`

Specifies whether to snap added or moved special wires to the closest routing track in the preferred direction for the layer automatically.
Default: `0`

Note: See `-snap_to_pin` for a description of the software's behavior when both `-snap_to_track` and `-snap_to_pin` are set to `1`.

`-snap_to_track_regular {0 | 1}`

Specifies whether to snap added or moved signal wires to the closest routing track in the preferred direction for the layer automatically.
Default: `0`

Note: See `-snap_to_pin` for a description of the software's behavior when both `-snap_to_track_regular` and `-snap_to_pin` are set to `1`.

`-spacing_horizontal` *`value`*

Specifies the distance between horizontal wires.

`-spacing_vertical` *`value`*

Specifies the distance between vertical wires.

`-split_wide_wires {0 | 1}`

Splits wires automatically when the width of a wire exceeds the `MAXWIDTH` parameter in the LEF file.
Default: `1`

`-stop_at_drc {0 | 1}`

Specifies whether the software can cause a DRC violation by moving or stretching a wire.

- Specify `1` to stop moving or stretching the wire at the closest location available before a violation occurs.
- Specify `0` to allow moving or stretching a wire to a location where a DRC violation occurs.

Note: If you specify `1`, you cannot move a wire to the desired destination if that wire passes through a location where a DRC violation occurs. Specify `0` to permit the wire to pass through to the desired location.
Default: `0`

`-stretch_end {high | low}`

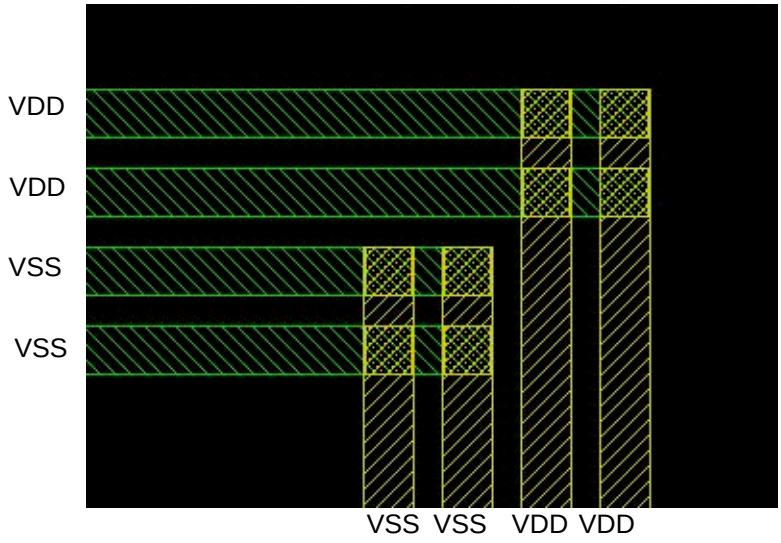
Specifies the direction in which to stretch or reduce wires.

`high`	Stretches or reduces wires from the top for vertical wires or from the right for horizontal wires.
`low`	Stretches or reduces wires from the bottom for vertical wires or from the left for horizontal wires.

`-use_wire_group {0 | 1}`

Groups multiple wires from the same net. Specify this parameter to decrease resistance and provide stronger connections.
Default: `0` (The wire editor does not use wire groups.)

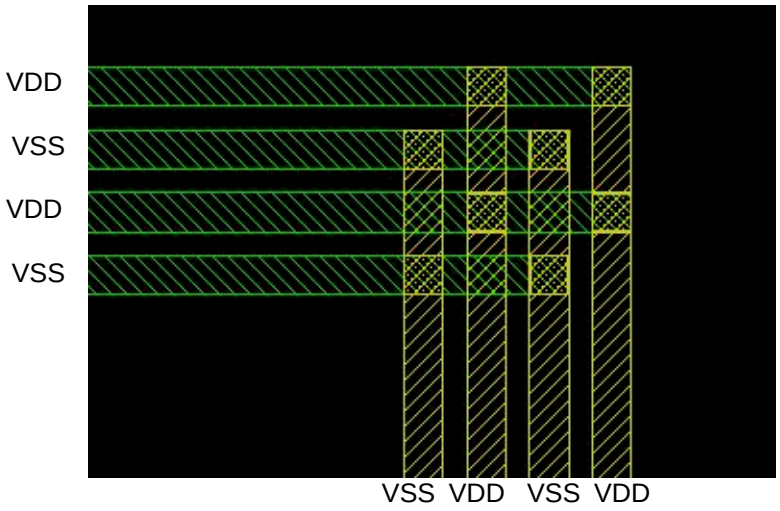
The following figure shows power and ground wires that form a wire group.



`-use_interleaving_wire_group {0 | 1}`

Alternates the specified wires in the wire group. For example, if you specify `-nets VDD VSS -use_interleaving_wire_groups 1`, the software alternates the power and ground wires in the wire group.
Default: `0`

When `-use_interleaving_wire_groups`, the software draws a wire group like the following:



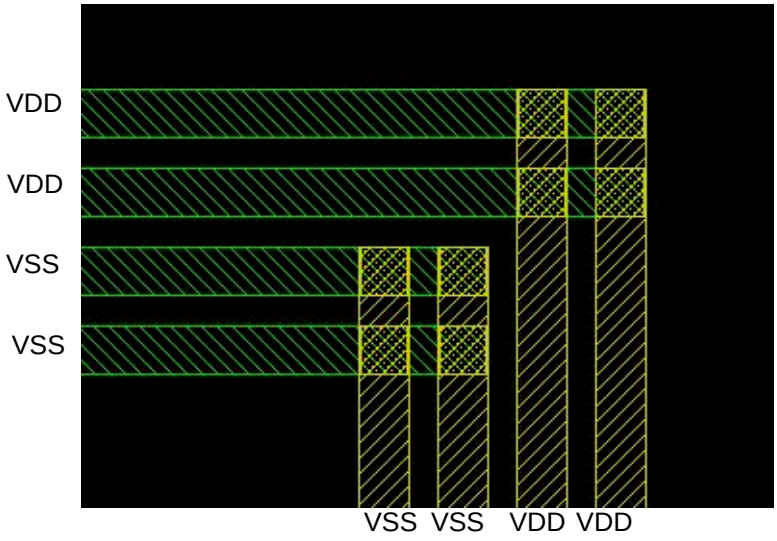
`-use_wire_group_bits` *`number_of_bits`*

Specifies the number of times to replicate a wire in the wire group. For example, if you specify VSS and VDD nets, and specify 2 for this parameter, the software creates a wire group with two power wires and two ground wires.
Default: `0` (disabled)

Important

The value of this parameter must be greater than 1 to enable wire groups.

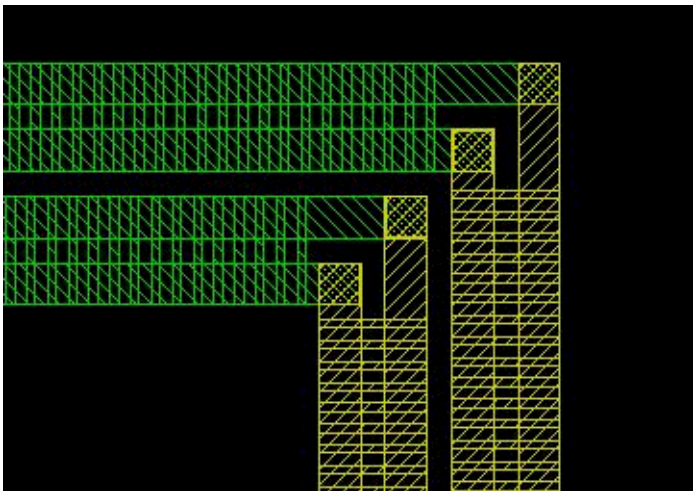
When `-use_wire_group_bits 2` is specified, the Encounter software draws a wire group like the following:



`-use_wire_group_reinforcement {0 | 1}`

Creates slots in the wire group by creating wires that connect the wires in the wire group, but are orthogonal to them. Slots help to reduce resistance in power networks. Specifying both this parameter and `-use_interleaving_wire_group` creates shorts in the wire group.
Default: `0`

When `-use_wire_group_reinforcement 1` is specified, the Encounter software draws a wire group like the following:



`-use_wire_group_reinforcement_spacing` *`spacing`*

Specifies the spacing for the orthogonal wires in the wire groups.

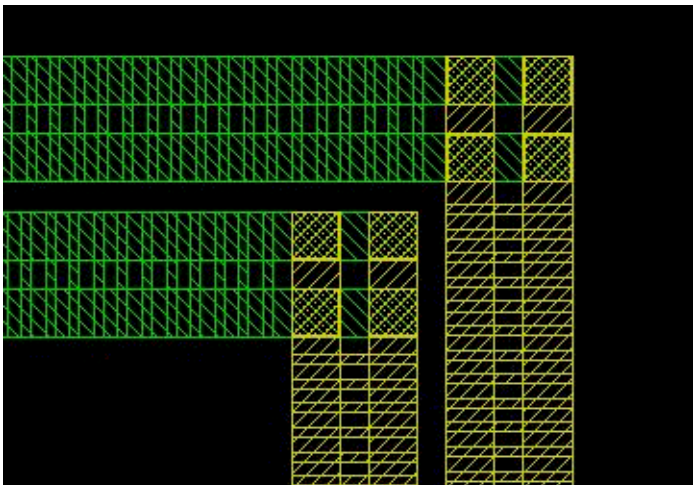
`-use_wire_group_reinforcement_width` *`width`*

Specifies the width of the orthogonal wires in the wire groups.

`-use_wire_group_reinforcement_group_via {0 | 1}`

Specifies that the wire group uses group vias. Group vias are connected vias in a wire group. Using group vias ensures that the contact is maintained if one of the vias in the wire group produces a void.
Default: `0`

In the following figure, the wire group has group vias:



`-width_horizontal` *`value`*

Specifies the width for horizontal wires.

`-width_vertical` *`value`*

Specifies the width for vertical wires.

setEditNetsFromBrowser

`setEditNetsFromBrowser`

Updates the Edit Route form with information from the Design Browser.

Parameters

None

setSpecialRouteOption

```
setSpecialRouteOption
     [-layer_minimum layerName]
     [-layer_maximum layerName]
     [-orthogonal_connection_only {0 | 1}]
     [-drc_on {0 | 1}]
```

Controls how to make connections and whether to perform DRC checking.

Parameters

`-drc_on {0 | 1}`

Specifies whether to check DRC rules during wire editing.

`-layer_maximum` *`layerName`*

Specifies the maximum layer for routing. Does not create wires above this layer.

`-layer_minimum` *`layerName`*

Specifies the minimum layer for routing. Does not create wires below this layer.

`-orthogonal_connection_only {0 | 1}`

Specifies whether to consider wires and pins that are in the same direction as a target for connection. For example, if you extend a wire to the left, any horizontal wire it meets is not considered for a target if the value of this parameter is set to `1`. However, a vertical wire is considered a target.
Default: `1`

setViaEdit

```
setViaEdit
     [-create_by {Geometry | Viacell}]
     [-auto_snap {0 | 1}]
     [-cut_layer cut_layer]
     [-x_space width]
     [-y_space length]
     [-cols integer]
     [-rows integer]
     [-x_size width]
     [-y_size length]
     [-x_topenc width]
     [-y_topenc length]
     [-x_botenc width]
     [-y_botenc length]
     [-viacell cellname]
```

Updates the Edit Via form, as well as vias that are subsequently created with the `editAddVia` command, with the information specified by the parameters.

Parameters

`-auto_snap {0 | 1}` Controls whether to snap vias to the intersection of wires on the same net or to the manufacturing grid.

- Specify `1` to snap the via to the center of the intersection of wires on the same net.
- Specify `0` to keep the via in the location where it is added or moved, snapping only to the manufacturing grid.
  Default: `1`

`-cols integer` Specifies the number of horizontal cuts in the via.
Default: `1`

`-create_by {Geometry | Viacell}`

Controls whether to create vias based on the `-viacell` parameter or to create vias based on the other parameters for `setViaEdit`.

- Specify `Viacell` to create the via with the same geometry as the cell specified by the `-viacell` parameter.
- Specify `Geometry` to create the via with the geometry specified by the other parameters in this command. For more information, see "-create_by Geometry" on page 2400.

Default: `Geometry`

`-cut_layer` *`cut_layer`*

Specifies the via layer name for the via to created or modify.
Default: `V12`

`-rows` *`integer`*
Specifies the number of vertical cuts in the via.
Default: `1`

`-x_botenc` *`width`*
Specifies the width in microns of the metal overhang for the layer at the bottom of the via.
Default: `0.2`

`-x_size` *`width`*
Specifies the width in microns of each cut.
Default: `0.2`

`-x_space` *`width`*
Specifies the horizontal distance in microns between via cuts.
Default: `0.2`

`-x_topenc` *`width`*
Specifies the length in microns of the metal overhang for the layer at the top of the via.
Default: `0.2`

`-viacell` *`cellname`*
Specifies the name of the via cell from the LEF or DEF file.

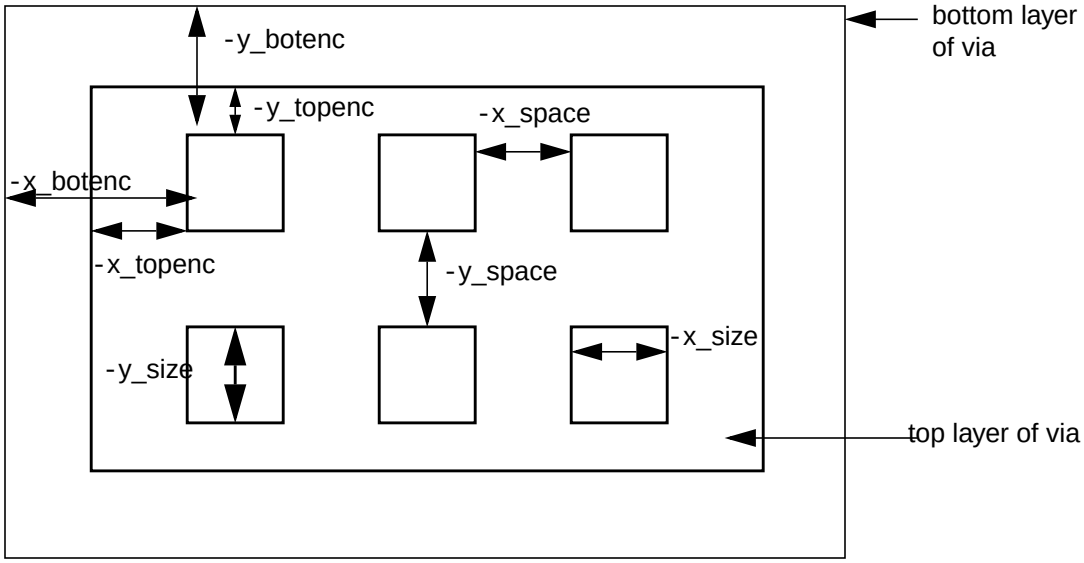
`-y_botenc` *`width`*
Specifies the length in microns of the metal overhang for the layer at the bottom of the via.
Default: `0.2`

`-y_size` *`width`*
Specifies the length in microns of each cut.
Default: `0.2`

`-y_space` *`width`*
Specifies the vertical distance in microns between via cuts.
Default: `0.2`

`-y_topenc` *`width`* Specifies the width in microns of the metal overhang for the layer at the top of the via.
Default: `0.2`

-create_by Geometry



Example

- The following commands add a via with two rows and three columns. The origin of the via is at coordinates `550`, `675`:

```
setViaEdit -Geometry -rows 2 -cols 3
editAddVia 550 675
```

42

Yield Analysis Commands

loadYieldTechFile

`loadYieldTechFile` *`fileName1 fileName2 fileName3 ...`*

Loads a technology-specific file or files that contain yield coefficients for short defects, open defects, via failure rates, and cell failure rates, as supplied by the fab or library vendor. All of the yield data can be supplied by one file, or it can be split among several files. Often the cell yield data is in a separate file from the via and routing yield data. The loadConfig command calls `loadYieldTechFile` during design import.

If no yield technology files exist, the Encounter software creates a default file using generic yield values suitable for experimentation with reportYield and yield optimization commands. To get accurate yield analysis, however, you must get realistic values from the fab or library vendor.

The files can optionally be encrypted using the `lib_encrypt` executable that is installed along with the Encounter software. For example:

`lib_encrypt` *`file`*`.dfy` *`file_encrypted`*`.yld`

The `loadYieldTechFile` command decrypts the input automatically.

For more information, including a description of the file format, see the "Analyzing Yield" chapter in the *Encounter User Guide*.

Parameters

`fileName`

Specifies the yield technology file or files. The last value read is used during cost calculations. For example, if cell A's cost is specified several times, in one or more file, the last one read is the one that is used.

The files can be encrypted and the command accepts a mixture of encrypted and non-encrypted files.

Related Topics

- Analyzing Yield chapter in the *Encounter User Guide*

readHif

```
readHif
     [help]
     -file filename
```

Reads the specified Hotspot Interface Format (HIF) file for the NanoRoute® router to use to fix lithography problems.

Use this command with `writeHif` for incremental lithography checking and fixing.

Note: `readHif` is aliased to `loadViolationReport`.

Parameters

`-file filename`	Specifies the HIF file to read.
`-help`	Outputs a brief description that includes type and default information for each `readHif` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man readHif`.

reportYield

```
reportYield
     [-createYieldFile newFileName |
      [-both]
      [-cell]
      [-critPathCellYield [-maxSlack value]]
      [-detailed]
      [-detailed_layer]
      [-effort {low| medium | high}]
      [-gridX value]
      [-gridY value]
      [-outfile fileName]
      [-shrink factor]
      [-tdsd]
      [-yld fileName1 fileName2 fileName3 ...] ]
```

Reports the expected defect-limited yield for the design using critical area analysis, based on the technology you are using and the probability of failure of the cells, vias, and point defects in routing. Uses technology-specific data from yield files supplied by the fabricator. Generates a yield report and a yield map, similar to a congestion map, that uses colors to indicate the yield loss graphically.

This command uses a stair-step approximation for 45-degree routes, and can analyze yield for wide-wire routes, as would occur for redistribution layer routing connected to bumps (RDL routing), but not for X-routing, where there are many minimum-width routes.

When you run this command with the `-createYieldFile` parameter, the software creates and loads a default yield file. When you run it after loading a yield file, it creates a yield map and report. When you save the design, the software saves the yield map. When you restore the design, the software restores the yield map.

The software can parse encrypted TSMC yield data files.

Parameters

`-both`
Specifies that `reportYield` adjust its critical area analysis calculations to account for the case where a single defect particle is large enough to cause both an open and a short. When `-both` is specified, `reportYield` does not double count the critical area for the large defects.

Check with your fab before using this parameter.

Classical critical analysis theory already accounts for large particles by having separate open and short defect densities. However, some fabs recommend using this parameter to match empirical results more closely.

`-cell`
Analyzes and reports yield for cells only. Use this parameter to speed up yield analysis by limiting it to cells, and skipping vias and routing.

`-createYieldFile` *`newFileName`*

Creates and loads a default Cadence yield file without running the `reportYield` command.

Use this parameter to create a file if one was not loaded with the `loadYieldTech` file command or imported by the `loadConfig` command. The next time you run `reportYield`, the software uses *`newFileName`* as the yield file.

`-critPathCellYield`

Reports cell costs on critical paths.

`-detailed`
Instructs the software to generate a detailed report. The detailed report contains the information included in the non-detailed report, plus detailed yield cost per cell and per location. For more information on the detailed report, and an example, see “Detailed Report” in the “Analyzing Yield” chapter of the *Encounter User Guide*.

`-detailed_layer` Further divides the subarea (location) data for via and routing costs from the detailed report into subarea per LEF routing layer.

If you specify both `-detailed` and `-detailed_layer`, `reportYield` generates a detailed report and a report for each layer.

`-gridX` *`value`* Specifies the step size of the yield map, in microns, in the horizontal direction.
Default: `50`

`-effort {low | medium | high}`

Specifies the effort level for critical area analysis. Typically, when you increase the effort level, the accuracy of the critical area analysis increases, but so does the run time.
Default: `low`
Specify one of the following values:

`low`	Computes interpolation points for critical area analysis, using a reasonable merge distance.
`medium`	Computes critical area analysis points in the yield file based on user-specified checkpoints, instead of using interpolation.
`high`	Computes each critical area analysis point in the yield file and makes the merge distance large enough to avoid merge errors, even for large defects.

`-gridY` *`value`* Specifies the step size of the yield map, in microns, in the horizontal direction.
Default: `50`

`-maxSlack` *`value`*

Specifies the threshold for maximum slack for a path to be considered as critical. The value for this parameter can be negative or positive.
Default: `0.0`

`-outfile` *`fileName`*

Specifies a name for the yield report. The report includes an overall yield result you can use to estimate the improvement in yield from different optimization techniques. If you specify this parameter, you must specify a filename for the report or the software will issue an error message and the command will exit.

If you run `reportYield` without specifying `-outfile`, the software generates a report named *`design_name`*`.dfm.rpt` in the current directory.

For more information on the report, and an example, see "Yield Report" in the "Analyzing Yield" chapter of the *Encounter User Guide*.

`-shrink` *`factor`*

Specifies a factor that allows you to use full-node size data for routing yield analysis on a half-node process by using an "optical shrink." The software uses the shrink factor to modify critical-area analysis to understand that the on-die geometry is a "shrink" of the drawn geometry, in order to compute the metal open and short yields from "shrunk" defect sizes correctly. For example, you might apply a shrink factor of `0.9` to scale yield analysis for a 90 nm design to a half-node of 80 nm.

Cell yield and via yield are not affected by the shrink (any changes to those values should already be in the `.yld` file data).

Note: If the TSMC defect density file specifies a shrink factor, the software ignores this parameter.

`-tdsd`

Calls the `tsmc2yld` utility to translate a TSMC defect parameter file to an encrypted Cadence `.yld` file.

You can use the `-tdsd` parameter to call the `tsmc2yld` utility, or you can run the `tsmc2yld` utility itself to translate the file. The utility is installed under the `tools/fe/bin` directory.

The `tsmc2yld` syntax is

```
tsmc2yld
  -dpFile defectParameterFile
  -techLef techLefFile.lef
  -yldFile fileName.yld
  [-cellCA cellCAAFile]
  [-key keyFile]
  [-quarter quarter]
```

Important

When you use `-tdsd` to call the utility, it uses the first three (required) parameters only.

`-cellCA` *`cellCAAFile`*

Specifies a cell yield library to use for critical area analysis, generally in an encrypted format. For information on the file syntax, see TSMC documentation.

`-dpFile` *`defectParameterFile`*

Specifies the defect parameter file from TSMC. The file may specify acceptable failure rates for eight consecutive quarter-year periods. By default, the software uses the first quarter's failure rate and generates an encrypted yield file based on that quarter, but you can specify a different quarter with the `-quarter` parameter.

`-key` *`keyFile`*

Specifies the vendor-provided key to use to decrypt the defect parameter file.

`-quarter` *`quarter`*

Specifies the quarter and year whose failure rate to use for calculations. Use one of the following formats to specify the quarter:

- *`YYYYQN`*, for example `2008Q3`
- *`YYQN`*, for example `08Q3`

`Q` must be an integer between 1 and 4.

`-techLef` *`techLefFile`*`.lef`

Specifies the technology LEF file. This file supplies the LEF layer names. Layer names can be specified by using one of the following formats: `M*X` or `m*X`, where `X` can be one or two digits. For example, `M12`, `metal3`, and `Met7` are all allowed.

Note: The `tsmc2yld` utility accepts comment lines beginning with #.

`-yldFile` *`fileName`*`.yld`

Specifies a name for the yield file in Cadence encrypted format. This is the file that the Encounter software uses during yield analysis.

`-yld` *`fileName1 fileName2 fileName3`*

Specifies a yield technology file or files.

The files specified by this parameter overwrite the files specified by the configuration file (`.conf`) and the `loadYieldTechFile` command.

Examples

The following command generates a non-detailed report named `mydesign.yld.rpt` for a design named `mydesign`:

```
reportYield
```

The following command generates a detailed report named `mydesign.yld.rpt` for a design named `mydesign`:

```
reportYield -detailed
```

The following command generates a non-detailed report named `ABC.yld.rpt:`

```
reportYield -outfile ABC.yld.rpt
```

The following command generates a detailed yield report named `ABC.detailed.yld.rpt`:

```
reportYield -outfile ABC.detailed.yld.rpt -detailed
```

The following command shows the usage of `tsmc2yld`:

```
tsmc2yld -dpfile CAA_defect_n65_08Q2.txt -techLef myTechLef.lef \
   -yldFile myYieldFile.yld -quarter 08Q2
```

Related Topics

- Analyzing Yield chapter in the *Encounter User Guide*

writeHif

```
writeHif
     [-help]
     -file filename
```

Writes out a HIF file after the NanoRoute router fixes lithography problems. The file reports the areas where the router made changes.

Use this command with `readHif` for incremental lithography checking and fixing.

Parameters

`-file` *`filename`*	Specifies the HIF file to write out.
`-help`	Outputs a brief description that includes type and default information for each `writeHif` parameter. For a detailed description of the command and all of its parameters, use the `man` command: `man writeHif`.

43

Basic Database Access Tcl Commands

dbGet

```
dbGet
     [-p | -p_number]
     [-u]
     [-regexp]
     [-d]
     {objectList | head | top | selected}
     [.object_type]*
     [.attribute_name | .? | .?? | .?h]
     [pattern]
```

Returns object and attribute information for the specified database object in the design. The `dbGet` command takes an object or object list as a starting point, then uses a period to traverse to other related objects, or to access attributes. Additional periods can be used to continue traversing the database schema.

For example, the following command returns the master cell names for all of the instances in the design:

```
dbGet top.insts.cell.name
```

- `top` is a pointer to the top cell.
- `insts` is a list of pointers to the instances in the top cell.
- `cell` is a list of pointers to the master cells for each instance.
- `name` is the master cell name.

The `dbGet` command is case insensitive. An empty field returns a value of `0x0`.

For a list of the object attributes that can be queried for information, or changed, see Appendix A, “Database Object Information.”

Parameters

`-p | -p_number`	Specifies the level to traverse back through the specified objects for the query. For example, if you specify `-p` or `-p1`, the software traverses one level back through the specified objects. If you specify `-p2`, the software traverses two levels back through the specified objects. The `-p` parameter is used with pattern name matching (*`pattern`*) to create a subset of names by which to query.

`-u` — Removes duplicate objects from the query results so that the results list only contains unique entries.

For information such as cell names, the results list can have the same entry multiple times. You can use the `-u` parameter to filter the results, so that the software only returns a unique list of different cell names.

`-regexp` — Uses regular expression pattern matching (Tcl `regexp` command).

Default: Uses simple wild card pattern matching

`-d` — Returns values in database integers.

Default: Returns values in floating point user units

`{`*`objectList`*` | head | top | selected}`

Specifies the starting point for the query.

- *`objectList`* — Specifies one or more object pointers as the starting point. You can specify a homogeneous or heterogeneous list.
- `head` — Specifies the pointer for the root, or head of the design as the starting point (also known as `dbgHead`).
- `top` — Specifies the pointer to the top cell in the design as the starting point (also known as `dbgTopCell`).
- `selected` — Specifies that the selected objects are the starting point.

`.`*`object_type`*`*` — Specifies the object type to query. You can specify more than one object in the query.

`{`*`attribute_name`*` | .? | .?? | .?h}`

Specifies the type of information to return.

`.attribute_name`

Returns the value of the specified object attribute.

`.?`	Returns a list of the available objects and attributes for the last object in the query. The software returns the object and attribute names only, not their values. For a list of objects, the field information is returned once for each type of object in the list. Most lists are of the same type; therefore, returning the same fields many times does not make sense.
`.??`	Returns a list of the objects and attributes, and their values, for the last object in the query. Use the `setDbGetMode` command to set controls for displaying `.??` query information.
`.?h`	Returns a list of the available objects and attributes for the last object in the query, and also includes the following information: ■ A short description for each object and attribute. ■ The type for each object. ■ The legal enum values for each attribute. ■ Whether the attribute value can be set (whether it is editable).

pattern

Specifies a string expression to use to match object or attribute names, or attribute values.

When you use *pattern* to match object or attribute names, the *pattern* is case insensitive. For example the following commands all return the instance placement status (`inst: pStatus`):

```
dbGet top.insts.? ps*
dbGet top.insts.? PS*
dbGet top.insts.? pS*
dbGet top.insts.? pstatus
dbGet top.insts.? PSTATUS
dbGet top.insts.? pStatus
```

When you use *pattern* to match attribute values (strings or enums), the *pattern* is case sensitive. For example, the following command returns the list of pointers for the instances that have `placed` status:

```
dbGet -p top.insts.pStatus placed
```

However, the following command returns `0x0`, because `PLACED` does not match the legal `placed` enum value:

```
dbGet -p top.insts.pStatus PLACED
```

Default: The software uses simple wildcard matching (`*`, `?`).

Examples

- The following command returns a list of pointers for the instances with names that match the `*reg*` pattern:

  ```
  dbGet top.insts.name *reg*
  ```

- The following command returns the names of all nets that start with the letter `I`. Specifying this command returns the same information as specifying the `dbFindNetsByName I*` command.

  ```
  dbGet top.nets.name I*
  ```

- The following command returns the names of all terminals that start with the letter `I`. Specifying this command returns the same information as specifying the `dbFindTermsByName I*` command.

  ```
  dbGet top.terms.name I*
  ```

- The following command returns a list of pointers to the instances that have a placement status of `Fixed`:

```
dbGet -p top.insts.pStatus fixed
```

- The following command returns a list of names for instances that have a placement status of `Fixed`:

  ```
  dbGet [dbGet -p top.insts.pStatus fixed].name
  ```

- The following commands also return a list of instance names for instances that have a placement status of Fixed:

  ```
  set fixedInst dbGet -p top.insts.pStatus fixed
  dbGet $fixedInsts.name
  ```

- The following command returns the status of the top cell in the design:

  ```
  dbGet top.?? status*
  ```

  The software displays the following information:

  ```
  statusClockSynthesized: 0
  statusGRouted: 0
  statusIoPlaced: 1
  statusPlaced: 1
  statusPowerAnalyzed: 0
  statusRCExtracted: 0
  statusRouted: 0
  statusScanOpted: 0
  ```

- The following command returns a list of the available objects and attributes for instances of the top cell:

  ```
  dbGet top.insts.?
  ```

  The software displays the following information:

  ```
  inst: box cell instTerms isDontTouch isHaloBlock isJtagElem isPhysOnly
  isSpareGate name objType orient pStatus pgCellTerms pgTermNets pt
  ```

- The following command displays the available objects and attributes, and their values, for the instances in the top cell. By default, the software displays information for the first 10 instances only. Use the `setDbGetMode` command to change the display limit.

  ```
  dbGet top.insts.??
  ```

  The software displays the following information:

  ```
  box: {314.16 302.4 316.8 307.44}
  cell: 0x12186fa8
  instTerms: 0x1330fa38 0x1330fa6c
  isDontTouch: 0
  isHaloBlock: 0
  isJtagElem: 0
  isPhysOnly: 0
  ```

```
isSpareGate: 0
name: i_10181
objType : inst
orient : MX
pStatus : placed
pgCellTerms: 0x13811088 0x138110cc
pgTermNets: 0x0
pt: {314.16 302.4}

box: {258.72 282.24 261.36 287.28}
cell: 0x12186c38
instTerms: 0x1330faa0 0x1330fad4
isDontTouch: 0
isHaloBlock: 0
isJtagElem: 0
isPhysOnly: 0
isSpareGate: 0
name: i_10177
objType : inst
orient : MX
pStatus : placed
pgCellTerms: 0x13810ef0 0x13810f34
pgTermNets: 0x0
pt: {258.72 282.24}

...
```

- The following command returns a list of the available objects and attributes for instances of the top cell. The software also displays a short description for each object and attribute, the type of each object, the legal enum values for each attribute, and whether the attribute value can be set by the user.

  ```
  dbGet top.insts.?h
  ```

  The software displays the following information:

  ```
  ==========================================================================
  inst: Instance - canonical (flat), equivalent to DEF COMPONENT. Points to a
  libCell or ptnCell
  ==========================================================================
  box: rect, Bounding box of instance
  cell: obj(libCell ptnCell), Pointer to child cell master or ptnCell
  instTerms: objList(instTerm), List of pointers to instance terminal
  isDontTouch: bool, Indicates that the instance is marked Don’t Touch
  ```

```
isHaloBlock: bool, Indicates that the instance is a halo block
isJtagElem: bool, Indicates that the instance is a Jtag element
isPhysOnly: bool, Indicates that the instance is physicl only
isSpareGate: bool, Indicates that the instance is a spare gate
name: string, Fully qualified (path) name of the instance
objType: enum(bndry densityShape fPlan group hInst hInstTerm hNet hTerm head
inst instTerm layer libCell net netGroup pBlkg pd pinGroup prop ptnCell rBlkg
sNet sWire shape stdRow term topCell vCell via wire), Object type: Instance
orient(settable): enum(MX MX90 MY MY90 R0 R180 R270 R90 Unknown), Instance
placement orientation
pStatus(settable): enum(cover fixed placed unplaced), Instance placement
status
pgCellTerms: objList(term), List of pointers to the instance’s power terminals
pgTermNets: objList(net), List of pointers to nets attached to P/G terminals
pt(settable): pt, Location of the instance
```

dbSchema

```
dbSchema
     [-help]
     [objNamePattern [objAttrNamePattern]]
```

Returns all of the available objects and attributes for the specified database object, and also includes the following information:

- A short description for each object and attribute.
- The type for each object.
- The legal enum values for each attribute.
- Whether the attribute value can be set (whether it is editable).

You can specify an object name, or a simple search pattern to match object names. If you specify `dbSchema`, the software returns a list of the database object names. If you specify `dbSchema *`, the software returns information for all database objects. All query information is output in the log file.

The `dbSchema` command also obeys the `setDbGetMode -displayFormat` setting to determine whether information is returned in simple or table format.

You can use the `dbSchema` command to return information on any database object listed in Appendix A, "Database Object Information."

Note: The information returned by the `dbSchema` command is equivalent to that returned by the `dbGet` command with the `.?h` operator; however, the `dbSchema` command does not require the object you query to actually exist in the database. For example, if you want to query related objects and attributes for a placement blockage, but do not have any placement blockages in the design, you can use the `dbSchema` command, but not the `dbGet` command.

Parameters

`-help`	Outputs a brief description that includes type and default information for each `dbSchema` parameter
`objNamePattern`	Specifies the object for which to return the information. You can specify an object name (such as `inst`, or `term`), or a string expression to use to match object names. The software uses simple wildcard matching (`*`, `?`).

`objAttrNamePattern` Specifies a secondary pattern for filtering object, or obejct attributes for the specified objects. This is similar to the `.?h [pattern]` filtering in the `dbGet` command.

Examples

- The following command returns the available net attributes that begin with `is`:

```
dbSchema net is*

================================================================================
net: Canonical (flat) net (equivalent to connectivity in DEF NETS and
SPECIALNETS
================================================================================
isClock: bool, Indicates that net is a clock net
isExternal: bool, Indicates that net is connected to a terminal
isFixedBump: bool, Indicates that the net is connected to a bump
isGnd: bool, Indicates that net is ground
isPatternTrunk: bool, Indicates that the net is routed with a trunk pattern
isPhysOnly: bool, Indicates that the net is physical only
isPwrOrGnd: bool, Indicates that net is power or ground
isScanNet: bool, Indicates that net is a scan net
```

- The following command returns all objects that contain the attribute `pStatus`:

```
dbSchema * pStatus
================================================================================
inst: Instance - canonical (flat), equivalent to DEF COMPONENT. Points to a
libCell or ptnCell.
================================================================================
pStatus(settable): enum(cover fixed placed unplaced), Instance placement
status
================================================
term: Terminal for libCell, ptnCell, or topCell
================================================
pStatus(settable): enum(cover fixed placed unplaced), Placement Status of the
terminal
```

- The following command returns the available objects and attributes for an instance:

```
dbSchema inst
================================================================================
```

```
inst: Instance - canonical (flat), equivalent to DEF COMPONENT. Points to a
libCell or ptnCell.
================================================================================
box: rect, Bounding box of the instance
cell: obj(libCell ptnCell), Pointer to child cell master or ptnCell
instTerms: objList(instTerm), List of pointers to instance terminals
isDontTouch: bool, Indicates that the instance is marked Don't Touch
isHaloBlock: bool, Indicates that the instance is a halo block
isJtagElem: bool, Indicates that the instance is a Jtag element
isPhysOnly: bool, Indicates that the instance is physical only
isSpareGate: bool, Indicates that the instance is a spare gate
name: string, Fully qualified (path) name of the instance
objType: enum(bndry densityShape fPlan group hInst hInstTerm hNet hTerm head
inst instTerm layer libCell net netGroup pBlkg pd prop ptnCell rBlkg row sWire
shape term topCell vCell via wire), Object type: Instance
orient(settable): enum(MX MX90 MY MY90 R0 R180 R270 R90 Unknown), Instance
placement orientation
pStatus(settable): enum(cover fixed placed unplaced), Instance placement
status
pgCellTerms: objList(term), List of pointers to the instance's power terminals
pgTermNets: objList(net), List of pointers to nets attached to P/G terminals
pt(settable): pt, Location of the instance
```

- The following command returns the available objects and attributes for database objects that start with the letter p:

```
 bdschema p*
================================================
pBlkg: Placement blockage (hard, soft, partial)
================================================
attr(settable): enum(inst pushdown undefined), Indicates whether the blockage
has been pushed down or is assoicated with an instance.
name(settable): string, Name of partial blockage
objType: enum(bndry densityShape fPlan group hInst hInstTerm hNet hTerm head
inst instTerm layer libCell net netGroup pBlkg pd prop ptnCell rBlkg row sWire
shape term topCell vCell via wire), Object type: Placement blockage
shapes: objList(densityShape shape), List of pointers to shapes (hard/soft) or
densityShapes (partial)
type(settable): enum(hard partial soft), Blockage type (hard, soft, partial)
================
pd: Power domain
================
core2Bot(settable): coord, Distance between the power domain edge and it's core
box
```

```
core2Left(settable): coord, Distance between the power domain edge and it's
core box
core2Right(settable): coord, Distance between the power domain edge and it's
core box
core2Top(settable): coord, Distance between the power domain edge and it's core
box
extBot(settable): coord, Maximum search distance for power connections
extLeft(settable): coord, Maximum search distance for power connections
extRight(settable): coord, Maximum search distance for power connections
extTop(settable): coord, Maximum search distance for power connections
gapBot(settable): coord, Minimum spacing to other power domains or rows
gapLeft(settable): coord, Minimum spacing to other power domains or rows
gapRight(settable): coord, Minimum spacing to other power domains or rows
gapTop(settable): coord, Minimum spacing to other power domains or rows
group: obj(group)
isAlwaysOn(settable): bool, Indicates that the power domain is always on
isDefault: bool, Indicates that the power domain is the default power domain
name: string, Name of the power domain
objType: enum(bndry densityShape fPlan group hInst hInstTerm hNet hTerm head
inst instTerm layer libCell net netGroup pBlkg pd prop ptnCell rBlkg row sWire
shape term topCell vCell via wire), Object type: Power domain
==============
prop: Property
==============
name: string, Property name
objType: enum(bndry densityShape fPlan group hInst hInstTerm hNet hTerm head
inst instTerm layer libCell net netGroup pBlkg pd prop ptnCell rBlkg row sWire
shape term topCell vCell via wire), Object type: Property
parent: obj(parent), Pointer to the parent object referencing the property
value: value, Property value (depends on valueType)
valuetype: enum(Double bits box integer loc pointer string), Type of value
stored in the property (int, float, etc.)
=====================================================
ptnCell: Partition cell, created by partition command
=====================================================
name: string, Name of cell
numBidirs: int, Number of bidir terminals in the cell
numInputs: int, Number of input terminals in the cell
numPGTerms: int, Number of power/ground terminals in the cell
numRefs: int, Number of times cell is used in design
numTerms: int, Number of terminals in the cell
objType: enum(bndry densityShape fPlan group hInst hInstTerm hNet hTerm head
inst instTerm layer libCell net netGroup pBlkg pd prop ptnCell rBlkg row sWire
shape term topCell vCell via wire), Object type: Partition cell
```

```
pgTerms: objList(term), List of power/ground terminals in the cell
props: objList(prop), List of pointers to properties
terms: objList(term), List of pointers to terminals in the cell
```

- The following command returns the available object and attributes for all types of blockages:

```
dbSchema *Blkg


================================================
pBlkg: Placement blockage (hard, soft, partial)
================================================
attr(settable): enum(inst pushdown undefined), Indicates whether the blockage
has been pushed down or is assoicated with an instance.
name(settable): string, Name of partial blockage
objType: enum(bndry densityShape fPlan group hInst hInstTerm hNet hTerm head
inst instTerm layer libCell net netGroup pBlkg pd prop ptnCell rBlkg row sWire
shape term topCell vCell via wire), Object type: Placement blockage
shapes: objList(densityShape shape), List of pointers to shapes (hard/soft) or
densityShapes (partial)
type(settable): enum(hard partial soft), Blockage type (hard, soft, partial)
=======================
rBlkg: Routing blockage
=======================
attr(settable): enum(default exceptPGNet fills inst pushdown slots), Routing
blockage qualifier (inst, pushdown, etc.)
layer: obj(layer), Pointer to layer of blockage
name(settable): string, Name of routing blockage
objType: enum(bndry densityShape fPlan group hInst hInstTerm hNet hTerm head
inst instTerm layer libCell net netGroup pBlkg pd prop ptnCell rBlkg row sWire
shape term topCell vCell via wire), Object type: Routing blockage
shapes: objList(shape), List of pointers to shapes that define the blockage
area
```

- The following commands return all available objects and attributes for a via in table format:

```
setDbGetMode -displayFormat table
dbSchema via


================================================
via: Via cell (equivalent to LEF VIA or DEF VIA)
================================================
botLayer                          | obj(layer), Pointer to bottom routing layer
botRects                          | rect, List of rectangles (typically only
one) on bottom layer
cutLayer                          | obj(layer), Pointer to cut layer
```

```
cutRects                          | rect, List of rectangles on cut layer
isDefault                         | bool, Indicates that the via is a default
via (LEF VIA DEFAULT)
isNonDefault                      | bool, Indicates that the via is declared in
a LEF NONDEFAULTRULE
name                              | string, Via name
objType                           | enum(bndry densityShape fPlan group hInst
hInstTerm hNet hTerm head inst instTerm layer libCell net netGroup pBlkg pd
prop ptnCell rBlkg row sWire shape term topCell vCell via wire), Object type:
Via cell
topLayer                          | obj(layer), Pointer to top routing layer
topRects                          | rect, List of rectangles (typically only
one) on top routing layer
```

dbSet

```
dbSet
     [-d]
     {object | objectList | head | top | selected}[.objectType]*.attributeName
         attributeValue
```

Changes the specified attribute value for a database object. The `dbSet` command takes an object or object list as a starting point, then uses a period to traverse to other related objects, or access attributes. Additional periods can be used to continue traversing the database schema.

The `dbSet` command does not create or delete objects.

Use the `dbGet` command with the `.?h` operator, or see Appendix A, "Database Object Information", to find out which object attributes can be set.

Parameters

`-d`	Specifies values in database units, specified in integers. *Default:* Specifies values in user units, in microns.
`object`	Specifies an object pointer.
`objectList`	Specifies a list of object pointers. The list must be homogeneous.
`head`	Specifies the pointer for the root, or head of the design as the starting point (also known as `dbgHead`).
`top`	Specifies the pointer to the top cell in the design as the starting point (also known as `dbgTopCell`).
`selected`	Specifies that the selected object (or objects) is the starting point.
`.objectType`	Specifies the object type.
`.attributeName`	Specifies the object attribute to change.
`attributeValue`	Specifies the new attribute value.

Examples

- The following command changes the placement status of the top cell instances to Fixed:

  ```
  dbSet top.insts.pStatus fixed
  ```

dbTransform

```
dbTransform
     {-inst instPtr | -cell cellPtr -pt pt -orient orient}
     -localPt localPts
```

Takes the local coordinates of a cell and returns them in context with the global design space. You can specify a specific instance of a cell in the design, or you can specify a library cell, its location in the design, and instruct the software to perform a “what-if” translation of coordinates within that cell.

Parameters

`-cell` *`cellPtr`*	Specifies a pointer to a library cell. You must specify the `-pt` and `-orient` parameters with `-cell`.
`-inst` *`instPtr`*	Specifies a pointer to a specific instance/bump of a cell in the design.The software derives the coordinates and the orientation for the cell instance from the design.
`-localPt` *`localPts`*	Specifies the local coordinates to convert into global coordinates
`-orient` *`orient`*	Specifies the orientation of the cell specified with `-cell`. You must specify the `-cell` and `-pt` parameters with `-orient`. Valid orientation values are: `R0`, `R180`, `R90`, `R270`, `MY`, `MX`, `MX90`, `MY90`.
`-pt` *`pt`*	Specifies the lower left coordinate for the cell specified with `-cell`. You must specify the `-cell` and `-orient` parameters with `-pt`.

Examples

- Assume that a shape exists on *M1* within a NAND2 cell, from (`1 1`) to (`2 2`). If an instance of the NAND2 cell is placed at (`10 10`), with an `R0` (N) orientation, the following command converts the local points (`1 1 2 2`) of the NAND2 cell into their global coordinates:

  ```
  dbTransform -cell [dbGet -p head.allCells.name NAND2] /
                -pt {10 10} -orient R0 -localPt {1 1 2 2}
  ```

  The software returns the following information:

  ```
  {11 11 12 12}
  ```

- Assume that the NAND2 cell placed at (`10 10`), with an `R0` (N) orientation is named `i1`. The following command converts the local points (`1 1 2 2`) of `i1` into their equivalent global coordinates:

  ```
  dbTransform -inst [dbGet -p top.insts.name i1] -localPt {1 1 2 2}
  ```

  The software returns the following information:

  ```
  {11 11 12 12}
  ```

dbu2uu

```
dbu2uu
     [-unit number]
     {value | {value_list}}
```

Takes an arbitrary list of values and returns it in the same form, with each database integer value converted to its user unit floating point equivalent.

Parameters

`-unit number`

Specifies the convert factor by which to divide the database integer value.

Value: One of `100`, `200`, `1000`, `2000`, `10000`, or `20000`

Default: Uses the LEF `UNITS DATABASE MICRONS` value from the Encounter database technology file, or the OpenAccess equivalent.

`{value | {value_list}}`

Specifies the value or value list to convert. You can specify a single value, or an arbitrary list of values, including points, rectangles and coordinates.

Type: Integer

Examples

- The following command converts the database value `100` into its equivalent user unit value, based on a convert factor of `2000`:

  ```
  dbu2uu -unit 2000 100
  ```

  The software returns the following information:

  ```
  0.050
  ```

- The following command converts the database values for the list of numbers {1000 1000 2000 2000} into their equivalent user unit values, based on a convert factor of `1000`:

  ```
  dbu2uu -unit 1000 {1000 1000 2000 2000}
  ```

  The software returns the following information for the list of numbers. The list is implied, but not shown.

  ```
  1.000 1.000 2.000 2.000
  ```

getDbGetMode

```
getDbGetMode
     [-quiet]
     [-displayFormat]
     [-displayLimit]
```

Displays the following information about a `setDbGetMode` parameter in the Encounter log file and in the Encounter console:

- Parameter name
- Current value
- Type (Boolean, string, and so on)
- Whether the current value was set by user

If you do not specify a parameter, the software displays information for all of the `setDbGetMode` parameters.

Parameters

`parameter_names`	Displays information for the specified parameters. You can specify one or more parameters. See `setDbGetMode` for descriptions of the parameters you can specify.
`-quiet`	Displays the current settings for the specified parameters in Tcl list format only. If you specify `-quiet` without any parameters, the software displays the current settings of all `setDbGetMode` parameters in Tcl list format.

Examples

- The following command displays the current setting of the `-displayFormat` parameter:

  ```
  getDbGetMode -displayFormat
  ```

  The software displays the following information:

  ```
  -displayFormat simple      # enums={simple table}, default=simple
  simple
  ```

- The following command displays the current settings for all `setDbGetMode` parameters:

  ```
  getDbGetMode
  ```

  The software displays the following information:

  ```
  -displayFormat simple      # enums={simple table}, default=simple
  -displayLimit 10           # int, default=10

  {displayFormat simple} {displayLimit 10}
  ```

- The following command displays the current setting of the `-displayFormat` parameter in Tcl list only:

  ```
  getDbGetMode -displayFormat -quiet
  ```

  The software displays the following information:

  ```
  simple
  ```

- The following command displays the current settings for all `setDbGetMode` parameters in Tcl list format only:

  ```
  getDbGetMode -quiet
  ```

  The software displays the following information:

  ```
  {displayFormat simple} {displayLimit 10}
  ```

setDbGetMode

```
setDbGetMode
     [-help]
     [-reset]
     [-displayFormat {simple | table}]
     [-displayLimit value]
```

Controls certain aspects of how the dbGet command displays object information.

Use the getDbGetMode command to display the current settings for the setDbGetMode command.

Parameters

`-displayFormat {simple | table}`

Determines whether to output the results of a .?? query as a simple list, or in a table.

Default: simple

`-displayLimit` *value*

Specifies the maximum number of objects, or object attributes to display in the output for a .?? query.

Default: 10

`-help` Outputs a brief description that includes type and default information for each setDbGetMode parameter.

`-reset` Resets parameters to their default values. The -reset parameter *must* be the first parameter specified. If you specify -reset by itself, the software resets all setDbGetMode parameters to their default values. If you specify parameters after -reset, the software resets only those parameters to their default values.

Examples

- The following command instructs the dbGet command to output the results of a .?? query as a table.

  ```
  setDbGetMode -displayformat table
  ```

  Query results are displayed in the following format:

```
attribute/object                | value
--------------------------------------------------------------------------------
-------------------------------------------
box                             | {0 0 1.32 5.04}
cell                            | 0x121ede48
instTerms                       | 0x13450290 0x134502c4
isDontTouch                     | 0
isHaloBlock                     | 0
isJtagElem                      | 0
isPhysOnly                      | 0
isSpareGate                     | 0
name                            | IOPADS_INST/PKS_Opt_2126
objType                         | inst
orient                          | R0
pStatus                         | unplaced
pgCellTerms                     | 0x13a3db98 0x13a3dbdc
pgTermNets                      | 0x131aa5c8 0x131aa688
pt                              | {0 0}
```

- The following command resets the `-displayformat` parameter to its default value:

  ```
  setDbGetMode -reset -displayFormat
  ```

  Query results are displayed in a simple list format:

  ```
  box: {0 0 1.32 5.04}
  cell: 0x121ede48
  instTerms: 0x13450290 0x134502c4
  isDontTouch: 0
  isHaloBlock: 0
  isJtagElem: 0
  isPhysOnly: 0
  isSpareGate: 0
  name: IOPADS_INST/PKS_Opt_2126
  objType : inst
  orient : R0
  pStatus : unplaced
  pgCellTerms: 0x13a3db98 0x13a3dbdc
  pgTermNets: 0x131aa5c8 0x131aa688
  pt: {0 0}

  box: {0 0 1.32 5.04}
  cell: 0x121ede48
  ```

```
instTerms: 0x134502f8 0x1345032c
isDontTouch: 0
isHaloBlock: 0
isJtagElem: 0
isPhysOnly: 0
isSpareGate: 0
name: IOPADS_INST/PKS_Opt_2125
objType : inst
orient : R0
pStatus : unplaced
pgCellTerms: 0x13a3db98 0x13a3dbdc
pgTermNets: 0x131aa5c8 0x131aa688
pt: {0 0}
```

- The following command resets all `setDbGetMode` parameters to their default values:

  ```
  setDbGetMode -reset
  ```

uu2dbu

```
uu2dbu
     [-unit number]
     {value | {value_list}}
```

Takes an arbitrary list of values and returns it in the same form, with each user unit floating point value converted to its database integer value equivalent.

Parameters

`-unit` *number*
: Specifies the convert factor by which to multiply the user unit value.

  Value: One of `100`, `200`, `1000`, `2000`, `10000`, or `20000`

  Default: Uses the LEF `UNITS DATABASE MICRONS` value from the Encounter database technology file, or the OpenAccess equivalent.

{*value* | {*value_list*}}
: Specifies the value or value list to convert. You can specify a single value, or an arbitrary list of values, including points, rectangles and coordinates.

  Type: Float

Examples

- The following command converts the user unit value `30` into its equivalent database integer value, based on a convert factor of `2000`:

  ```
  uu2dbu -unit 2000 30
  ```

  The software returns the following information:

  ```
  60000
  ```

- The following command converts the user unit values for the list of numbers {`30 40`} into their equivalent database unit values, based on a convert factor of `200`:

  ```
  uu2dbu -unit 200 {30 40}
  ```

  The software returns the following information for the list of numbers. The list is implied, but not shown.

  ```
  6000 8000
  ```

A

Configuration File Variables

The following table lists the `rda_Input()` variables that are saved in the default configuration file.

Related Topics

- Inputs in the *Encounter Flat Implementation Flow Guide*

Table A-1 Configuration File Variables

Variable Name	Description	Example
`rda_Input(ui_netlist)`	Specifies the gate-level Verilog netlist files to import.	`set rda_Input(ui_netlist) {z650n3_n2n.enc.dat/z650n3.v}`
`rda_Input(ui_netlisttype)`	Specifies the source of the design netlist. You can specify either `{Verilog}` or `{OA}`.	`set rda_Input(ui_netlisttype){Verilog}`
`rda_Input(ui_settop)`	Specifies whether the top cell name is automatically assigned `{0}`, or is set by the user `{1}`.	`set rda_Input(ui_settop){0}`
`rda_Input(ui_topcell)`	Specifies the top cell name.	`set rda_Input(ui_topcell){TOP}`
`rda_Input(ui_timelib,min)`	Specifies the min timing libraries to import.	`set rda_Input(ui_timelib,min){fast.lib}`

Variable Name	Description	Example
`rda_Input(ui_timelib,max)`	Specifies the max timing libraries to import.	`set rda_Input(ui_timelib,max) {slow.lib}`
`rda_Input(ui_timelib)`	Specifies the common timing libraries to import. The libraries s are used for both min and max path analysis.	`set rda_Input(ui_timelib){tdsp_core.lib}`
`rda_Input(ui_leffile)`	Specifies LEF files to import.	`set rda_Input(ui_leffile){technology.lef stdcell.lef}`
`rda_Input(ui_timingcon_file)`	Specifies the Synopsys Design Constraint (SDC) timing constraint file to import in regular non-ILM flow.	`set rda_Input(ui_timingcon_file) {dtmf_chip.pt}`
`rda_Input(ui_iolib)`	Specifies the I/O library files to import. .	`set rda_Input(ui_iolib){dtmf_chip.io}`
`rda_Input(ui_areaiolib)`	Specifies the area I/O library files to import.	`set rda_Input(ui_areaiolib){tsmc_areaio}`
`rda_Input(ui_io_file)`	Specifies the I/O constraint file.	`set rda_Input(ui_io_file){dtmf.io}`
`rda_Input(ui_exc_net)`	Specifies the hierarchical nets to be excluded from delay calculation, and assigned default delay values.	`set rda_Input(ui_exc_net){netA netB}`
`rda_Input(ui_delay_limit)`	Sets the minimum pin count number for a net that is excluded from delay calculaton.	`set rda_Input(ui_delay_limit){1000}`

Variable Name	Description	Example
`rda_Input(ui_net_delay)`	Sets the delay value for a net that meets the pin limit default.	`set rda_Input(ui_net_delay){1000.0ps}`
`rda_Input(ui_net_load)`	Sets the load for a net that meets the pin limit default.	`set rda_Input(ui_net_load){0.5pf}`
`rda_Input(ui_in_tran_delay)`	Sets the default input transition delays, and the ideal clock transition time for delay calculation.	rda_Input(ui_in_tran_delay) {120.0ps}
`rda_Input(ui_ilmdir)`	Specifies the directory from which to read the ILM files.	set rda_Input(ui_ilmdir) {newILM.dir}
`rda_Input(ui_timingcon_file,full)`	Specifies the timing constraint file to import in the ILM flow.	set rda_Input(ui_timingcon_file,full) {dtmf_chip.sdc}
`rda_Input(ui_cts_cell_list)`	Specifies a list of buffer and inverter names for CTS.	rda_Input(ui_cts_cell_list) {TB2INVCLXN TB2BUFCLXR}
`rda_Input(ui_oa_designLib)`	Specifies the OpenAccess database library.	set rda_Input(ui_oa_designLib) {myOALib}
`rda_Input(ui_oa_designCell)`	Specifies the OpenAccess database cell.	set rda_Input(ui_oa_designCell) {myOACell}
`rda_Input(ui_oa_designView)`	Specifies the OpenAccess database view.	set rda_Input(ui_oa_designView) {myOAView}

Variable Name	Description	Example
rda_Input(ui_oa_reflib)	Specifies the OpenAccess reference libraries to import. Each reference library is processed using the abstract view name list (ui_oa_abstractname). For example, if the reference library is "lib1 lib2", and the abstract view name list is "abstract abstract2", LEF MACRO information is processed for lib1 with the abstract view. Then, for any cells in lib1 that didn notvhave abstract, but did have abstract2, that view is processed for MACRO information. If a cell has both views, the first one is used. The process then is repeated for lib2.	rda_Input(ui_oa_reflib) {OAREFLIB1 OAREFLIB2}
rda_Input(ui_oa_abstractname)	Specifies the OpenAccess view names that the software should examine to find the equivalent LEF MACRO information (for example, PINS, OBS, FOREIGN). If this value is null (""),the software uses abstract as the value.	rda_Input(ui_oa_abstractname) {ABS1}
rda_Input(ui_oa_layoutname)	Specifies the layout view names to import.	rda_Input(ui_oa_layoutname) {LAY3}
rda_Input(ui_pwrnet)	Specifies power nets used in the design.	rda_Input(ui_pwrnet) {VDD}

Variable Name	Description	Example
`rda_Input(ui_gndnet)`	Specifies ground nets used in the design.	`set rda_Input(ui_gndnet) {VSS}`
`rda_Input(ui_captbl_file)`	Specifies the typical, best, and worst case capacitance tables to import.	set rda_Input(ui_captbl_file) {-worst TC280C410_Tj125.captbl -best TC280C410_Tj27.captbl -typical TC280C410_Tj29.captbl }
`rda_Input(ui_defccap_scale)`	Sets the default scaling values to multiply the extracted interconnect capacitance value to a new scaled value.	rda_Input(ui_defcap_scale) {1.0}
`rda_Input(ui_detcap_scale)`	Sets the detail scaling values to multiply the extracted interconnect capacitance value to a new scaled value.	rda_Input(ui_detcap_scale) {1.0}
`rda_Input(ui_xcap_scale)`	Sets the scaling values to multiply the extracted interconnect coupling capacitance value to a new scaled value.	rda_Input(ui_xcap_scale) {1.0}
`rda_Input(ui_res_scale)`	Sets the scaling value to multiply the extracted interconnect resistance value to a new scaled value.	rda_Input(ui_res_scale) {1.0}
`rda_Input(ui_shr_scale)`	Specifies the value by which the software shrinks the design.	rda_Input(ui_shr_scale) {1.0}

Variable Name	Description	Example
`rda_Input(ui_rel_c_thresh)`	Specifies the relative capacitance threshold value.	rda_Input(ui_rel_c_thresh) {0.03}
`rda_Input(ui_tot_c_thresh)`	Specifies the total capacitance threshold value.	rda_Input(ui_tot_c_thresh) {5.0}
`rda_Input(ui_qxtech_file)`	Specifies the name of a QRC technology file.	rda_Input(ui_qxtech_file) {icecaps.tch}
`rda_Input(ui_qxlayermap_file)`	Specifies the layer map file used by both CCE extraction and QRC standalone sign-off RC extraction.	rda_Input(ui_qxlayermap_file) {lefdef_layer.map.txt}
`rda_Input(ui_qxlib_file)`	Specifies the path to the optional cell library created with LibGen.	rda_Input(ui_qxlib_file) {tsmc18_6lm.cl}
`rda_Input(ui_qxconf_file)`	Specifies the name of a user-created QRC configuration file that contains additional commands to use during QRC extraction.	rda_Input(ui_qxconf_file) {dtmf.qxconf}
`rda_Input(ui_rtllist)`	Specifies the RTL netlist files to import.	set rda_Input(ui_rtllist) {myRTLNetlist}
`rda_Input(ui_cdb_file,min)`	Specifies the noise library (`.cdB`) files used in hold analysis mode.	rda_Input(ui_cdb_file,min) {fast_2005.cdB}
`rda_Input(ui_cdb_file,max)`	Specifies the noise library (`.cdB`) files used in setup analysis mode.	rda_Input(ui_cdb_file,max) {slow_2005.cdB}
`rda_Input(ui_cdb_file)`	Specifies the names of the common noise library (`.cdB`) files used in CeltIC NDC analysis.	rda_Input(ui_cdb_file) {2005.cdB}

Variable Name	Description	Example
`rda_Input(ui_echo_file,min)`	Specifies the ECHO model files used in hold analaysis mode.	rda_Input(ui_echo_file,min) {dsp_echo.min}
`rda_Input(ui_echo_file,max)`	Specifies the ECHO model files used in hold analaysis mode.	rda_Input(ui_echo_file,max) {dsp_echo.max}
`rda_Input(ui_echo_file)`	Specifies the names of the common ECHO noise model files used in CeltIC NDC analysis.	rda_Input(ui_echo_file) {dsp_echo.com}
`rda_Input(ui_xtwf_file)`	Specifies the cross talk timing window file in ILM analysis.	rda_Input(ui_xtwf_file) {dtmf.xtwf}
`rda_Input(ui_smodDef)`	Specifies the Stamp model definition files.	set rda_Input(ui_smodDef) {wsg_icp.top.mod}
`rda_Input(ui_smodData)`	Specifies the Stamp model data files.	set rda_Input(ui_smodData) {wsg_icp.top.data}
`rda_Input(ui_locvlib)`	Specifies the LOCV library file.	set rda_Input(ui_locvlib) {dtmf_core.locv}
`rda_Input(ui_cap_unit)`	Specifies the capacitive load unit for objects specific to capacitive load.	rda_Input(ui_cap_unit) {pf}
`rda_Input(ui_time_unit)`	Specifies the time unit to be used for timing constraints.	rda_Input(ui_time_unit) {none}
`rda_Input(ui_yldfile)`	Specifies a yield file to import.	`set rda_Input(ui_yldfile) {coeff_cell}`
`rda_Input(ui_view_definition_file)`	Specifies a Tcl command script that contains the analysis view configurations for multi-mode multi-corner timing analysis and optimization.	set rda_Input(ui_view_definition_file) {mmmc.tcl}

Variable Name	Description	Example
`rda_Input(ui_dpath)`	Specifies the data path.	set rda_Input(ui_dpath) {}
`rda_Input(ui_scheduling_file)`	Specifies scheduling files.	set rda_Input(ui_scheduling_file) {scheduling_file}
`rda_Input(ui_kboxlib)`	Specifies the black box library files to import.	set rda_Input(ui_kboxlib) {tsmc_blackbox}
`rda_Input(ui_latency_file)`	Specifies the clock latency files.	set rda_Input(ui_latency_file) {latency_file.sdc}
`rda_Input(ui_core_cntl)`	Specifies how the core dimension of the chip is determined. You can specify either `{aspect}` or `{size}`.	rda_Input(ui_core_cntl) {aspect}
`rda_Input(ui_aspect_ratio)`	Specifies the aspect ratio.	rda_Input(ui_aspect_ratio) {1.0}
`rda_Input(ui_isOrigCenter)`	Specifies whether the origin of the floorplan should be at the center `{1}`, or at the lower left corner `{0}`.	rda_Input(ui_isOrigCenter) {0}
`rda_Input(ui_core_util)`	Specifies the utilization value for the core.	rda_Input(ui_core_util) {0.7}
`rda_Input(ui_core_height)`	Specifies the height of the core box. Specify this variable when the `ui_core_cntl` variable is `{size}`.	rda_Input(ui_core_height) {809.64}
`rda_Input(ui_core_width)`	Specifies the width of the core box. Specify this variable when the `ui_core_cntl` variable is `{size}`.	rda_Input(ui_core_width) {809.58}

Variable Name	Description	Example
`rda_Input(ui_core_to_left)`	Specifies the distance from the outside edge of the core box to the die edge on the left side.	rda_Input(ui_core_to_left) {20.42}
`rda_Input(ui_core_to_right)`	Specifies the distance from the outside edge of the core box to the die edge on the right side.	rda_Input(ui_core_to_right) {20.0}
`rda_Input(ui_core_to_top)`	Specifies the distance from the outside edge of the core box to the die edge on the top side.	rda_Input(ui_core_to_top) {20.0}
`rda_Input(ui_core_to_bottom)`	Specifies the distance from the outside edge of the core box to the die edge on the bottom side.	rda_Input(ui_core_to_bottom) {20.36}
`rda_Input(ui_max_io_height)`	Specifies whether the maximum I/O height `{1}`, or the minimum I/O height `{0}` should be used to calclulate the die size.	rda_Input(ui_max_io_height) {0}
`rda_Input(ui_row_height)`	Sets the standard cell row height.	rda_Input(ui_row_height) {5.04}
`rda_Input(ui_isVerticalRow)`	Specifies that the rows in the floorplan are vertical (`1`), instead of horizontal (`0`). **Note:** Support for vertical rows is a beta feature. Usage and support of this beta feature are subject to prior agreement with Cadence. Contact your Cadence representative if you have any questions.	rda_Input(ui_isVerticalRow) {0}

Variable Name	Description	Example
`rda_Input(ui_isHorTrackHalfPitch)`	Determines whether the horizontal track information used is from the DEF file.	rda_Input(ui_isHorTrackHalfPitch) {0}
`rda_Input(ui_isVerTrackHalfPitch)`	Determines whether the vertical track information used is from the DEF file.	rda_Input(ui_isVerTrackHalfPitch) {1}
`rda_Input(ui_ioOri)`	Specifies the orientation of the I/O cell, on the bottom edge of the design. Orientation can be `R0` (default), `R90`, `R180`, or `R270`. Orientation of remaining sides rotates by 90 degrees in the counterclockwise direction.	rda_Input(ui_ioOri) {R0}
`rda_Input(flip_first)`	Specifies the orientation of the bottom row in the core area. {1} specifies that the first row flips from the bottom up; {0} specifies that the second row flips from the bottom up.	`set rda_Input(flip_first) {1}`
`rda_Input(double_back)`	Determines whether the row orientations alternate between N {1} and FS {0}.	`set rda_Input(double_back) {1}`
`rda_Input(use_io_row_flow)`	Determines whether I/O rows are used for I/O placement {1}, or not {0}.	`set rda_Input(use_io_row_flow) {0}`
`rda_Input(assign_buffer)`	Specifies settings equivalent to the `setDoAssign` command. `{0}` equals `setDoAssign off`; `{1}` equals `setDoAssign on`.; `{1 -buffer` *`name`*`}` equals `setDoAssign on -buffer` *`name`*.	`set rda_Input(assign_buffer) {0}`

Variable Name	Description	Example
`rda_Input(import_mode)`	Specifies parameters equivalent to the `setImportMode` command parameters.	`set rda_Input(import_mode) {-treatUndefinedCellAsBbox 0 -keepEmptyModule 1 -useLefDef56 1}`

B

Database Object Information

Database Objects	Description
bndry (Boundary)	Boundary for placement control (fence, region, guide, etc).
cellDensity (Cell Density)	cellDensity.
densityShape (Density Shape)	Rectangle with metal/cut or placement density information.
fPlan (Floorplan)	Floorplan headers with pointers to floorplan objects.
group	Group of hInsts, insts, or groups.
head	Root/Head of the database.
hInst (Hierarchical Instance)	Hierarchical instance (derived from netlist).
hInstTerm (Hierarchical Instance Terminal)	Hierarchical instance terminal (seen from level above).
hNet (Hierarchical Net)	Hierarchical net (derived from netlist).
hTerm (Hierarchical Terminal)	Hierarchical terminal (seen from level below).
inst (Instance)	Instance - canonical (flat), equivalent to DEF COMPONENT. Points to a libCell or ptnCell.
instTerm (Instance Terminal)	Instance terminal (used in flattened connectivity).
layer	Layer (from technology source, LEF or OpenAccess).
libCell (Library Cell)	Library cell (from LEF or OpenAccess).

marker	DRC Marker.
net	Canonical (flat) net (equivalent to connectivity in DEF NETS and SPECIALNETS).
netGroup (Net Group)	Net group.
pBlkg (Placement Blockage)	Placement blockage (hard, soft, partial).
pd (Power Domain)	Power domain.
prop (Property)	Property.
ptnCell (Partition Cell)	Partition cell, created by partition command.
rBlkg (Routing Blockage)	Routing blockage.
row	Row (core), constructed from sites (equivalent to DEF ROWS).
rule	Rule information (equivalent to LEF/DEF NONDEFAULTRULE)
shape	Shape.
shapeVia (Via Shape)	layer Shape via.
site	site.
sWire (Special Wire)	Special wire (equivalent to DEF SPECIALNETS wiring).
term (Terminal)	Terminal for libCell, ptnCell, or topCell.
text	Text.
topCell (Top Cell)	Top cell, container for flattened connectivity.
vCell (Verilog Cell)	Intermediate module cell (equivalent to intermediate module cells in Verilog or OpenAccess module domain).
via	Via cell (equivalent to LEF VIA or DEF VIA).
wire	Wire (symbolic wire type, equivalent to DEF NETS wiring). Can represent a wire segment, a via, or both.

Attribute Location Abbreviations

Abbreviation	Actual Definition
coord (coordinate)	Single value; either int (`dbGet -d`), or float.
pt (point)	{*coord coord*}
ptList (list of points)	{*pt pt ...*} or
	{{*coord coord*} {*coord coord*}}
box	{*coord coord coord coord*}
rect (rectangle)	{*coord coord coord coord*}

Note: Outermost curly braces ({}) are implied in the return values for lists.

Supported Database Object Attributes

Note: For distance attributes, values are returned as floats when using user units, and as integers (int) when using database units (DBUs).

bndry (Boundary)

Child Object or Attribute	Type	Editable	Description
`boxes`	list (rect)	No	List of rectangles that define the boundary.
`hInst`	obj (hInst)	No	Pointer to parent hierarchical instance (if boundary is for a partition).
`objType`	enum	No	Object type: Boundary for placement control. Legal enum: `bndry`

Child Object or Attribute	Type	Editable	Description
`type`	enum	Yes	`fence`: Exclusive placement boundary. `region`: Inclusive placement boundary. `guide`: Suggested placement boundary. `cluster`: Keep group members near each other (boxes field is empty). `none`: Undefined. Legal enum: `fence`, `region`, `cluster`, `guide`, `none`.

cellDensity (Cell Density)

Child Object or Attribute	Type	Editable	Description
`layer`	obj (layer)	No	Pointer to layer object for density shapes.
`objType`	enum	No	Object type: Equivalent to LEF `MACRO DENSITY`. Legal enum: `density`
`shapes`	objList (densityShape)	No	List of pointers to densityShape objects.

densityShape (Density Shape)

Child Object or Attribute	Type	Editable	Description
`density`	float	No	Density value. Legal range: `0.0` - `1.0`
`objType`	enum	No	Object type: Rectangle with metal or placement density. Legal enum: `densityShape`

Child Object or Attribute	Type	Editable	Description
`rect`	rect	No	Rectangle. For cellDensity, the coordinates are local to cell, not relative to the design.

fPlan (Floorplan)

Child Object or Attribute	Type	Editable	Description
`bndrys`	objList (bndry)	No	List of pointers to boundaries (fence, region, etc.).
`box`	rect	No	Rectangle that defines the design size.
`coreBox`	rect	No	Rectangle that defines the core row area.
`groups`	objList (group)	No	List of pointers to groups.
`ioBox`	rect	No	Rectangle that defines the I/O area.
`netGroups`	objList (netGroup)	No	List of pointers to net groups.
`numRows`	int	No	Number of rows within the core area.
`objType`	enum	No	Object type: Floorplan, has pointers to most floorplan objects. Legal enum: `fPlan`
`pBlkgs`	objList (pBlkg)	No	List of pointers to placement blockages.
`rBlkgs`	objList (rBlkg)	No	List of pointers to routing blockages.
`rows`	objList (row)	No	List of pointers to rows.
`topCell`	obj (topCell)	No	Pointer to cell containing the floorplan data.

group

Child Object or Attribute	Type	Editable	Description
`conType`	enum	Yes	Constraint type for the group. Legal enum: `cluster`, `fence`, `guide`, `none`, `region`
`density`	float	No	Group density. Legal range: `0` - `100`
`members`	objList (hInst, inst, group)	No	List of pointers to group's members.
`name`	string	No	Name of group.
`objType`	enum	No	Object type: Group of hInsts, insts, or groups. Legal enum: `group`
`parent`	obj (group)	No	Pointer to the parent group, if object is a sub-group.
`pd`	object (pd)	No	Pointer to power domain (if group is power domain).
`props`	objList (prop)	No	List of pointers to properties.

head

Child Object or Attribute	Type	Editable	Description
`allCells`	objList (libCell, topCell, vCell, ptnCell)	No	List of pointers to cells of all types (library cells and design cells).
`dbUnits`	int	No	Database units per user unit.
`layers`	objList (layer)	No	List of pointers to layers.
`mfgGrid`	coord	No	Manufacturing grid.

Child Object or Attribute	Type	Editable	Description
objType	enum	No	Object type: Root/head of the database. Legal enum: head
props	objList (prop)	No	List of pointers to properties.
ptns	objList (ptns)	No	List of pointers to partitions in the design.
rules	objList (rule)	No	List of pointers to non-default rule objects.
vias	objList (rule)	No	List of a via master.

hInst (Hierarchical Instance)

Child Object or Attribute	Type	Editable	Description
allInsts	objList (hInst, inst)	No	List of pointers to all instances and hierarchical instances at the current level of the hierarchical instance.
allTreeInsts	objList (hInst inst)	No	List of pointers to all instances and hierarchical instances at the levels below the current hierarchical instance.
bndry	obj (bndry)	No	Pointer to the boundary.
group	obj (group)	No	Pointer to the parent group.
hInstTerms	objList (hInstTerm)	No	List of pointers to the hierarchical terminals.
hNets	obj (hNet)	No	List of pointers to hierarchical nets in the hierarchical instance.
name	string	No	Fully qualified (path) name of the hierarchical instance. Legal value: dbmHinstName
objType	enum	No	Object type: Hierarchical instance. Legal enum: hInst
parent	obj (topCell)	No	Pointer to the top cell of the design.

Child Object or Attribute	Type	Editable	Description
`props`	objList (prop)	No	List of pointers to properties.
`ptn`	obj (ptn)	No	Pointer to the partition.
`vCell`	obj (vCell)	No	Pointer to module cell object. that corresponds to the hierarchical instance.

hInstTerm (Hierarchical Instance Terminal)

Child Object or Attribute	Type	Editable	Description
`downHNet`	obj (hNet)	No	Pointer to the hierarchical net below.
`hInst`	obj (hInst)	No	Pointer to the hierarchical instance.
`hTerm`	obj (hTerm)	No	Pointer to the hierarchical terminal.
`layer`	obj (layer)	No	Pointer to the layer of the hierarchical instance terminal.
`name`	string	No	Fully qualified (path) name of the hierarchical instance terminal.
`net`	obj (net)	No	Pointer to the associated canonical (flat) net.
`objType`	enum	No	Object type: Hierarchical instance terminal. Legal enum: `hInstTerm`
`props`	objList (prop)	No	List of pointers to properties.
`pt`	pt	Yes	Location of the hierarchical instance terminal.
`term`	obj (term)	No	Pointer to terminal.
`upHNet`	obj (hNet)	No	Pointer to the hierarchical net above.

hNet (Hierarchical Net)

Child Object or Attribute	Type	Editable	Description
`allTerms`	objList (hInstTerm, hTerm, instTerm)	No	List of pointers to connections on the net.
`isGnd`	bool	No	Indicates that the net is a ground net.
`isPwrOrGnd`	bool	No	Indicates that the net is a power or ground net.
`name`	string	No	Net name (local).
`objType`	enum	No	Object type: Hierarchical net. Legal enum: `hNet`
`props`	objList (prop)	No	List of pointers to properties.

hTerm (Hierarchical Terminal)

Child Object or Attribute	Type	Editable	Description
`downHNet`	obj (hNet)	No	Pointer to the hierarchical net below.
`hInst`	obj (hInst)	No	Pointer to the hierarchical instance containing the hierarchical terminal.
`name`	string	No	Terminal name (local).
`net`	obj (net)	No	Pointer to the associated canonical (flat) net.
`objType`	enum	No	Object type: Hierarchical terminal. Legal enum: `hTerm`
`props`	objList (prop)	No	List of pointers to properties.
`upHNet`	obj (hNet)	No	Pointer to the hierarchical net above.

inst (Instance)

Child Object or Attribute	Type	Editable	Description
box	rect	No	Bounding box of instance.
cell	objList (libCell, ptnCell)	No	Pointer to child cell master or partition cell.
instTerms	objList (instTerm)	No	List of pointers to instance terminals.
isDontTouch	bool	No	Indicates that the instance is marked Don't Touch.
isHaloBlock	bool	No	Indicates that the instance is a halo block.
isJtagElem	bool	No	Indicates that the instance is a Jtag element.
isPhysOnly	bool	No	Indicates that the instance is physical only.
isSpareGate	bool	No	Indicates that the instance is a spare gate.
name	string	No	Fully qualified (path) name of the instance.
objType	enum	No	Object type: Instance. Legal enum: inst
orient	enum	Yes	Instance placement orientation.
pgCellTerms	objList (term)	Yes	List of pointers to instance's power terminals.
pgTermNets	objList (net)	No	List of pointers to nets attached to power/ ground terminals.
pHaloBot	coord	Yes	Specifies extra spacing around the inst that should not be used for placement.
pHaloBox	rect	Yes	Bounding box for the inst placement halo.
pHaloLeft	coord	Yes	Specifies extra spacing around the inst that should not be used for placement.
pHaloRight	coord	Yes	Specifies extra spacing around the inst that should not be used for placement.
pHaloTop	coord	Yes	Specifies extra spacing around the inst that should not be used for placement.

Child Object or Attribute	Type	Editable	Description
pStatus	enum (cover fix placed unplaced)	Yes	Instance placement status. Legal enum: cover, fixed, placed, unplaced.
pt	pt	Yes	Location of the instance.
rHaloBotLayer	obj (layer)	Yes	Specifies the bottom layer for which routing halo will be created.
rHaloSideSize	coord	Yes	Specifies routing halo around the inst.
rHaloTopLayer	obj (layer)	Yes	Specifies the bottom layer for which routing halo will be created.

instTerm (Instance Terminal)

Child Object or Attribute	Type	Editable	Description
cellTerm	obj (term)	No	Pointer to the equivalent cell terminal.
inst	obj (inst)	No	Pointer to the instance containing the instance terminal.
isInput	bool	No	Indicates that the terminal is an input.
isOutput	bool	No	Indicates that the terminal is an output.
isTieHi	bool	No	Indicates that the terminal is a tieHi.
isTieLo	bool	No	Indicates that the terminal is a tieLo.
layer	obj (layer)	No	Pointer to the layer of the hierarchical instance terminal.
name	string	No	Fully qualified instance terminal name, including the instance path.
net	obj (net)	No	Pointer to the net connected to the instance terminal.
objType	enum	No	Object type: Instance terminal. Legal enum: instTerm

Child Object or Attribute	Type	Editable	Description
pt	pt	No	Location of instance terminal (yellow square in display).

layer

Child Object or Attribute	Type	Editable	Description
area	float	No	Layer minimum area from LEF or OpenAccess.
backside	bool	No	Indicates that the layer is a backside layer (on the underside of the die).
densityStepX	coord	No	Layer density window step from LEF or OpenAccess.
densityStepY	coord	No	Layer density window step from LEF or OpenAccess.
densityWindowX	coord	No	Layer density window length from LEF or OpenAccess.
densityWindowY	coord	No	Layer density window length from LEF or OpenAccess.
direction	enum	No	Layer preferred direction from LEF or OpenAccess. Legal enum: `Diag135`, `Diag45`, `HVUnassigned`, `Horizontal`, `Vertical`.
extName	string	No	Name of the layer from LEF or OpenAccess.
fillActiveSpacing	coord	No	Layer fill minimum spacing from LEF or OpenAccess.
fillGapSpacing	coord	No	Layer fill to fill spacing from LEF or OpenAccess.
maxDensity	float	No	Layer fill maximum density from LEF or OpenAccess.

Child Object or Attribute	Type	Editable	Description
`maxWidth`	coord	No	Layer maximum wire width from LEF or OpenAccess.
`minDensity`	float	No	Layer fill minimum density from LEF or OpenAccess.
`minSpacing`	coord	No	Layer minimum spacing from LEF or OpenAccess.
`minWidth`	coord	No	Layer minimum wire width from LEF or OpenAccess.
`name`	string	No	Name of the layer.
`objType`	enum	No	Object type: Layer. Legal enum: `layer`
`offsetX`	coord	No	Layer offset X from LEF or OpenAccess.
`offsetY`	coord	No	Layer offset Y from LEF or OpenAccess.
`pitchX`	coord	No	Layer wire pitch X from LEF or OpenAccess
`pitchY`	coord	No	Layer wire pitch Y from LEF or OpenAccess.
`props`	objList (prop)	No	List of pointers to properties.
`spacingTables`	list	No	List of spacing table in LEF format.
`type`	enum	No	Type of layer. Legal enum: `MIMCap`, `TSV`, `cut`, `implant`, `invalid`, `masterslice`, `nwell`, `overlap`, `pwell`, `passivation`, `routing`.
`width`	coord	No	Layer wire width from LEF or OpenAccess.

libCell (Library Cell)

Child Object or Attribute	Type	Editable	Description
`allObstructions`	objList(cellDensity)	No	List of pointers to layerShapes and shapeVias that define cell obstruction geometries.
`baseClass`	enum (variant of LEF classes without the options)	No	Class type: Derived from LEF or OpenAccess (refer to documentation for complete mapping). Legal enum: `none`, `cover`, `block`, `pad`, `core`, `corner`.
`cellDensities`	objList(cellDensity)	No	List of pointers to cell density information. Equivalent to LEF `MACRO DENSITY`.
`eeqCells`	objList(libCell)	No	List of pointers to electrically equivalent cells. Equivalent to LEF `MACRO EEQ`
`foreigns`	objList(foreign)	No	List of pointers to the foreign references. Equivalent to LEF `MACRO FOREIGN`.
`isSequential`	bool	No	Indicates that the cell is sequential.
`name`	string	No	Name of the cell.
`numBidirs`	int	No	Number of bidirectional terminals in the cell.
`numInputs`	int	No	Number of input terminals in the cell.
`numOutputs`	int (derived value)	No	Number of output terminals in the cell.
`numPGTerms`	int	No	Number of power/ground terminals in the cell.
`numRefs`	int	No	Number of times cell is used in the design.
`numTerms`	int	No	Number of terminals in the cell.
`objType`	enum	No	Object type: Library cell. Legal enum: `libCell`
`pgTerms`	objList (term)	No	List of pointers to the power/ground terminals in the cell.

Child Object or Attribute	Type	Editable	Description
`props`	objList (prop)	No	List of pointers to properties.
`site`	obj(site)	No	Pointer to site of the cell.
`size`	pt	No	Size of the cell.
`subClass`	enum (variant of LEFF CLASS combinations, similar to OA's cellTypes)	Yes	Class sub type: Derived from LEF or OpenAccess (refer to documentation for complete mapping). Legal enum: `none`, `cornerTopLeft`, `cornerTopRight`, `cornerBottomLeft`, `cornerBottomRight`, `cover`, `coverBump`, `block`, `blockRing`, `blockBlackBox`, `blockSoft`, `pad`, `padInput`, `padOutput`, `padInout`, `padSpacer`, `padAreaIO`, `core`, `coreEndCapPre`, `coreEndCapPost`, `coreFeedthru`, `coreTieHigh`, `coreTieLow`, `coreSpacer`, `coreAntenna`, `coreWellTap`.
`symmetryR90`	bool	No	Indicates that the symmetry of the cell is R90.
`symmetryX`	bool	No	Indicates that the symmetry of the cell is X.
`symmetryY`	bool	No	Indicates that the symmetry of the cell is Y.
`terms`	objList (term)	No	List of pointers to signal terminals in the cell.

marker

Child Object or Attribute	Type	Editable	Description
`box`	Rect	No	Bounding box for marker.
`layer`	ob (layer)	No	Pointer to layer.
`message`	string	No	DRC marker message.
`messageid`	int	No	DRC marker message ID.

objType	enum	No	Object type: DRC marker.
originator	enum	No	Marker originator.
polyPts	list (pt)	No	Polygon boundry for the marker, the first point is not repeated as the last point in the list.
subType	enum	No	Marker subtype.
type	enum	No	Marker type: ACLimit, Antenna, Connectivity, Density, Electrical Floorplan, Geometry, IRDrop, MixedSignal, NRLitho, None, Overlap, Placement, Xtalk.

net

Child Object or Attribute	Type	Editable	Description
allTerms	objList (term, instTerm)	No	List of pointers to connections (terminals and instance terminals).
box	rect	No	Bounding box of net's wiring, if routed. Otherwise, bounding box of the terminals/ instance terminals on the net.
isClock	bool	No	Indicates that the net is a clock net.
isExternal	bool	No	Indicates that the net is connected to a terminal.
isFixedBump	bool	No	Indicates that the net is connected to a bump.
isGnd	bool	No	Indicates that the net is ground.
isPatternTrunk	bool	No	Indicates that the net is routed with a trunk pattern.
isPhysOnly	bool	No	Indicates that the net is physical only.
isPwrOrGnd	bool	No	Indicates that the net is power or ground.
isScanNet	bool	No	Indicates that the net is a scan net.

Child Object or Attribute	Type	Editable	Description
name	string	No	Canonical (flat) name of the net.
numInputTerms	int	No	Number of input terminals connected to the net.
numOutputTerms	int	No	Number of output terminals connected to the net.
numTerms	int	No	Number of terminals connected to the net.
objType	enum	No	Object type: Canonical (flat) net. Legal enum: net
props	objList (prop)	No	List of pointers to properties.
rule	obj(rule)	No	Non-default rule corresponding to the net. Nets with the default routing rule, will return NULL (0x0).
sVias	objList (sWire)	No	List of pointers to special vias (DEF SPECIALNETS equivalent).
sWires	objList (sWire)	No	List of pointers to special wires (DEF SPECIALNETS equivalent).
wires	objList (wire)	No	List of pointers to wires (DEF NETS equivalent).

netGroup (Net Group)

Child Object or Attribute	Type	Editable	Description
name	string	No	Group name.
nets	objList (net)	No	List of pointers to member nets.
objType	enum	No	Object type: Net group. Legal enum: netGroup

pBlkg (Placement Blockage)

Child Object or Attribute	Type	Editable	Description
`attr`	enum	Yes	Indicates whether the blockage has been pushed down, or is associated with an instance. Legal enum: `undefined`, `inst`, `pushdown`.
`density`	float	Yes	Density value of the blockage. Legal value: `0`, `5`, `10`, `15`, ...`100`
`name`	string	Yes	Name of blockage.
`objType`	enum	No	Object type: Placement blockage. Legal enum: `pBlkg`
`shapes`	objList (shape, densityShape)	No	List of pointers to shapes (hard/soft), or density shapes (partial).
`type`	enum	Yes	Type of placement blockage. Legal enum: `hard`, `soft`, `partial`.

pd (Power Domain)

Child Object or Attribute	Type	Editable	Description
`core2Bot`	coord	Yes	Distance between the power domain edge and its core box.
`core2Left`	coord	Yes	Distance between the power domain edge and its core box.
`core2Right`	coord	Yes	Distance between the power domain edge and its core box.
`core2Top`	coord	Yes	Distance between the power domain edge and its core box.
`extBot`	coord	Yes	Maximum search distance for power connections.

Child Object or Attribute	Type	Editable	Description
extLeft	coord	Yes	Maximum search distance for power connections.
extRight	coord	Yes	Maximum search distance for power connections.
extTop	coord	Yes	Maximum search distance for power connections.
gapBot	coord	Yes	Minimum spacing to other power domains or rows.
gapLeft	coord	Yes	Minimum spacing to other power domains or rows.
gapRight	coord	Yes	Minimum spacing to other power domains or rows.
gapTop	coord	Yes	Minimum spacing to other power domains or rows.
group	obj (group)	No	Pointer to the parent group object.
isAlwaysOn	bool	Yes	Indicates that the power domain is always turned on.
isDefault	bool	No	Indicates that the power domain is the default power domain.
name	string	No	Name of the power domain.
objType	enum	No	Object type: Power domain. Legal enum: pd

prop (Property)

Child Object or Attribute	Type	Editable	Description
name	string	No	Name of property.
objType	enum	No	Object type: Property. Legal enum: prop

Child Object or Attribute	Type	Editable	Description
parent	obj	No	Pointer to parent object referencing the property.
value		No	Property value (depends on valueType).
valueType	enum	No	Type of value stored in the property (int, float, etc.).

ptnCell (Partition Cell)

Child Object or Attribute	Type	Editable	Description
name	string	No	Name of the cell.
numBidirs	int	No	Number of bidirectional terminals in the cell.
numInputs	int	No	Number of input terminals in the cell.
numPGTerms	int	No	Number of power/ground terminals in the cell.
numRefs	int	No	Number of times the cell is used in the design.
numTerms	int	No	Number of terminals in the cell.
objType	enum	No	Object type.: Partition cell Legal enum: ptnCell
pgTerms	objList (term)	No	List of pointers to power/ground terminals in the cell.
props	objList (prop)	No	List of pointers to properties.
symmetryR90	bool	Yes	Indicates that the symmetry of the cell is R90.
symmetryX	bool	Yes	Indicates that the symmetry of the cell is X.
symmetryY	bool	Yes	Indicates that the symmetry of the cell is Y.
terms	objList (term)	No	List of pointers to terminals in the cell.

rBlkg (Routing Blockage)

Child Object or Attribute	Type	Editable	Description
attr	enum	Yes	Routing blockage qualifier. Legal enum: default, inst, pushdown, slots, fills, exceptPGNet
layer	obj (layer)	No	Pointer to layer of blockage.
name	string	Yes	Name of blockage.
objType	enum	No	Object type: Routing blockage. Legal enum: rBlkg
shapes	objList (shape)	No	List of pointers to shapes that define the blockage area.

row

Child Object or Attribute	Type	Editable	Description
box	rect	No	Bounding box of the row.
dir	enum	No	Indicates if the row is horizontal or vertical.
numX	int	No	Number of sites in X direction (refer to DEF ROW syntax).
numY	int	No	Number of sites in Y direction (refer to DEF ROW syntax).
objType	enum	No	Object type: Row (core). Legal enum: stdRow
orient	enum	No	Orientation of the sites in the row.
site	obj (site)	No	Pointer to site used in row.
stepX	coord	No	Step in X direction (refer to DEF ROW syntax).
stepY	coord	No	Step in Y direction (refer to DEF ROW syntax).

rule

Child Object or Attribute	Type	Editable	Description
`fromLib`	bool	No	Indicates whether the rule came from library technology or from the design. True: From LEF or OA tech. False: From DEF, OA database, or CreateNdr
`hardspacing`	bool	No	Indicates that any spacing values that exceed the LEF layer spacing requirements are ‘hard’ rules and not ‘soft’ rules.
`layerRule`	objList(layerRule)	No	List of pointers to layer cuts.
`minCuts`	list(list)	No	List of cut layer and minimum number of cuts allowed for any via using the specified cut layer.
`name`	string	No	Name of non-default rule.
objType	enum	No	Object Type: Rule Legal enum: `rule`
vias	objList(via)	No	List of via, default or USEVIA or derived from minicut

shape

Child Object or Attribute	Type	Editable	Description
`objType`	enum	No	Object type: Shape. Legal enum: `shape`
`path`	pts	No	Points of the path (if `type` equals `path`).
`poly`	pts	No	Points of the polygon (if `type` equals `poly`).
`rect`	rect	No	Bounding box of shape.

Child Object or Attribute	Type	Editable	Description
type	enum	No	Type of shape. Legal enum: path, rect, poly.

shapeVia (Via Shape)

Child Object or Attribute	Type	Editable	Description
loc	pt	No	Location of via.
objType	enum	No	Object type: Shape via. Legal enum: lShape
via	obj (via)	No	Pointer to the via master.

site

Child Object or Attribute	Type	Editable	Description
class	enum	No	Site Class (equivalent to LEF SITE CLASS). Legal enum: core, pad.
name	string	No	Name of site (equivalent to LEF SITE CLASS).
objType	enum	No	Object type: Site. Legal enum: site

sWire (Special Wire)

Child Object or Attribute	Type	Editable	Description
`beginExt`	coord	No	Extension of path at the first point (only on `path` type).
`box`	rect	No	Bounding box for shapes (actual shape if `geomType` equals `rect`).
`endExt`	coord	No	Extension of path at the last point (only on `path` type).
`geomType`	enum	No	Type of shape. Legal enum: `via`, `rect`, `poly`, `pathSeg`, `path`, `text`.
`layer`	obj (layer)	No	Pointer to layer (not used on `via` type).
`objType`	enum	No	Object type: Special wire. Legal enum: `sWire`
`polyPts`	ptList	No	Polygon boundary for the object, null for `via`, the first point is not repeated as the last point).
`pts`	ptList	No	1 point for `via`, 2 points for `pathSeg` center-line, n points for `path` center-line, n points for `poly`, null for `rect`.
`shape`	enum	No	LEF + SHAPE equivalents (ring, stripe, etc)
`shieldNet`	obj (net)	No	Pointer to net that is shielded if status is `shieldNet`.
`status`	enum	Yes	Wiring status (equivalent to DEF `SPECIALNETS` special wiring status). Legal enum: `cover`, `fixed`, `routed`, `shield`
`width`	coord	No	Width of `path` or `pathSeg` type.

term (Terminal)

Child Object or Attribute	Type	Editable	Description
antennas	objList(antenna)	No	List of pointers to antennas.
`cell`	obj (libCell, topCell, ptnCell)	No	Pointer to the cell (libCell, topCell, or ptnCell) that contains the terminal.
`inOutDir`	bool	No	
`isInput`	bool	No	Indicates that the terminal is an input.
`isOutput`	bool	No	Indicates that the terminal is an output.
`isScanClk`	bool	No	Indicates that the terminal is a scan clock terminal.
`isSpecial`	bool	No	Indicates that the terminal is Special.
`isTieHi`	bool	No	Indicates that the terminal is a tieHi.
`isTieLo`	bool	No	Indicates that the terminal is a tieLo.
`layer`	obj(term)	No	Pointer to layer.
`mustJoin`	obj(term)	No	Pointer to mustjoin terminal.
`name`	string	No	Terminal name.
`net`	obj (net)	No	Pointer to canonical (flat) net connected to the terminal.
`objType`	enum	No	Object type: Terminal. Legal enum: `term`
`orient`	enum	No	Orientation of the terminal.
`pins`	objList(pin)	No	List of pointers to the ports of the terminal. Equivalent to LEF `MACRO PIN PORT`.
`props`	objList(prop)	No	List of pointers to properties.
`pStatus`	enum	Yes	Placement Status of the terminal. Legal enum: `cover`, `fixed`, `placed`, `unplaced`.

Child Object or Attribute	Type	Editable	Description
shape	enum	No	Terminal shape. Legal enum: `abutment`, `feedThru`, `none`, `ring`.
`type`	enum	No	Terminal type. Legal enum: `analogTerm`, `asyncCtrlTerm`, `clockTerm`, `dQTerm`, `dTerm`, `fFQTerm`, `feedTerm`, `gatedClockTerm`, `groundTerm`, `latchQTerm`, `normalTerm`, `powerTerm`, `rSTerm`, `triStateTerm`.

text

Child Object or Attribute	Type	Editable	Description
`alignment`	enum	No	Horizontal and vertical alignment of the text setting. `enum:centerCenter, centerLeft, centerRight, lowerCenter, lowerLeft, lowerRight, upperCenter, upperLeft, upperRight.`
`drafting`	bool	No	Indicates if the text will always be displayed top-to-bottom or left-to-right. If this value is true the text will always remain readable even if the text is rotated or mirrored.
`font`	string		Text font.
`height`	coord	Yes	Text height.
`label`	string	Yes	Text string value.
`layer`	Obj (layer)	No	Pointer to layer.
`objType`	enum	No	Object type: Text. Legal enum: `text`
`orient`	enum	No	The orientation of the text. Legal enum: `MX MX90 MY MY90 R0 R180 R290 R90`

pt	pt	No	Location of the text.

topCell (Top Cell)

Child Object or Attribute	Type	Editable	Description
fPlan	obj (fPlan)	No	Pointer to the floorplan.
hInst	obj (hInst)	No	Pointer to the top-level hierarchical instance.
insts	objList (inst)	No	List of pointers to instances in the cell.
markers	objList(marker)	No	List of pointers to markers.
name	string	No	Name of the cell.
nets	objList (net)	No	List of pointers to canonical (flat) nets in the cell.
numBidirs	int	No	Number of bidirectional terminals in the cell.
numInputs	int	No	Number of input terminals in the cell.
numInputs	int	No	Number of input terminals in the cell.
numInsts	int	No	Number of instances in the cell.
numNets	int	No	Number of canonical (flat) nets in the cell.
numPhysInsts	int	No	Number of physical instances in the cell.
numPhysNets	int	No	Number of physical nets in the cell.
numTerms	int	No	Number of terminals in the cell.
objType	enum	No	Object type: Top cell. Legal enum: topCell
pgTerms	objList (term)	No	List of pointers to power/ground terminals in the cell.
physInsts	objList (inst)	No	List of pointers to physical instances in the cell.
physNets	objList (net)	No	List of pointers to physical nets in the cell.
props	objList (prop)	No	List of pointers to properties.

Child Object or Attribute	Type	Editable	Description
statusClockSynthehsized	bool	No	Design status: Clock synthesized.
statusGRouted	bool	No	Design status: Globally routed.
statusIoPlaced	bool	No	Design status: I/Os are Placed.
statusPlaced	bool	No	Design status: Placed.
statusPowerAnalyzed	bool	No	Design status: Power analyzed.
statusRCExtracted	bool	No	Design status: Parasitics extracted.
statusRouted	bool	No	Design status: Detail routed.
statusScanOpted	bool	No	Design status: Scan optimized.
symmetryR90	bool	Yes	Indicates that the symmetry of the cell is R90.
symmetryX	bool	Yes	Indicates that the symmetry of the cell is X.
symmetryY	bool	Yes	Indicates that the symmetry of the cell is Y.
terms	objList (term)	No	List of pointers to terminals in the cell.
texts	objList(text)	No	List of pointers to text.

vCell (Verilog Cell)

Child Object or Attribute	Type	Editable	Description
hInst	objList (hInst)	No	Pointer to the corresponding hierarchical instance. If the netlist data is not uniquified, the pointer will be to one of the hierarchical instances.
name	string	No	Name of the cell.
numOutputs	int	No	Number of output terminals in the cell.

Child Object or Attribute	Type	Editable	Description
objType	enum	No	Object type: Intermediate module cell. Legal enum: vCell
props	objList (prop)	No	List of pointers to properties.

via

Child Object or Attribute	Type	Editable	Description
botLayer	obj (layer)	No	Pointer to the bottom routing layer.
botRects	list (rect)	No	List of rectangles (typically only one) on bottom layer.
cutLayer	obj (layer)	No	Pointer to the cut layer.
cutRects	list (rect)	No	List of rectangles on cut layer.
isDefault	enum	No	Indicates that the via is a default via (LEF VIA DEFAULT).
isNonDefault	enum	No	Indicates that the via is declared in a LEF NONDEFAULTRULE statement.
name	string	No	Via name.
objType	enum	No	Object type: Via cell. Legal enum: via
topLayer	obj (layer)	No	Pointer to the top routing layer.
topRects	list (rect)	No	List of rectangles (typically only one) on top layer.

wire

Child Object or Attribute	Type	Editable	Description
beginExt	coord	No	Extension of wire at the first point.

Child Object or Attribute	Type	Editable	Description
endExt	coord	No	Extension of wire at the second point.
layer	obj (layer)	No	Pointer to layer of the wire (NULL if only via is present.
objType	enum	No	Object type: Wire. Legal enum: sWire
pts	ptList	No	2 points for the wire, NULL (0x0) if the sturcture only contains a via.
rule	obj(rule)	No	Non-default rule corresponding to the net. Nets with the default routing rule, will return NULL (0x0).
status	enum	Yes	Wiring status (equivalent to DEF NETS regular wiring status). Legal enum: cover, fixed, noshield, routed, unknown
via	obj (via)	No	Pointer to via cell. Returns NULL (0x0) if the structure only includes a wire segment and no via.
viaPt	pt	No	Location of via.
width	coord	No	Width of the wire (0 if only via is present).

Index

A

B

C

D

E

F

G

H

I

Q

R

S

T

U

V

W

Z