

PrimeTime®
Advanced Timing Analysis
User Guide

Version F-2011.06, June 2011

SYNOPSYS®

Copyright Notice and Proprietary Information

Copyright © 2011 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only. Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

“This document is duplicated with the permission of Synopsys, Inc., for the exclusive use of _____ and its employees. This is copy number _____.”

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Registered Trademarks (®)

Synopsys, AEON, AMPS, ARC, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, Chipidea, CHIPit, CODE V, CoMET, Confirma, CoWare, Design Compiler, DesignSphere, DesignWare, Eclipse, EMBED-IT!, Formality, Galaxy Custom Designer, Global Synthesis, HAPS, HapsTrak, HDL Analyst, HSIM, HSPICE, Identify, Leda, LightTools, MAST, MaVeric, METeor, ModelTools, NanoSim, NOVeA, OpenVera, ORA, PathMill, Physical Compiler, PrimeTime, SCOPE, SiVL, SNUG, SolvNet, Sonic Focus, STAR Memory System, Syndicated, Synplicity, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, the Synplicity logo, UMRBus, VCS, Vera, and YIELDExplorer are registered trademarks of Synopsys, Inc.

Trademarks (™)

AFGen, Apollo, ASAP, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Cosmos, CosmosLE, CosmosScope, CRITIC, CustomExplorer, CustomSim, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, DFTMAX, Direct Silicon Access, Discovery, Encore, EPIC, Galaxy, HANEX, HDL Compiler, Hercules, Hierarchical Optimization Technology, High-performance ASIC Prototyping System, HSIM^{plus}, i-Virtual Stepper, ICE, in-Sync, iN-Tandem, Intelli, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Macro-PLUS, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, ORAengineering, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, RippledMixer, Saturn, Scirocco, Scirocco-I, SiWare, Star-RCXT, Star-SimXT, StarRC, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCSi, VHDL Compiler, VMC, and Worksheet Buffer are trademarks of Synopsys, Inc.

Service Marks (SM)

MAP-in, SVP Café, and TAP-in are service marks of Synopsys, Inc.

SystemC is a trademark of the Open SystemC Initiative and is used under license.

ARM and AMBA are registered trademarks of ARM Limited.

Saber is a registered trademark of SabreMark Limited Partnership and is used under license.

All other product or company names may be trademarks of their respective owners.

Contents

What's New in This Release	xiv
About This User Guide	xiv
Customer Support.	xvi
1. Overview of Advanced Topics	
Topics Covered in This Manual.	1-2
2. Tcl Advanced Features	
Tcl Packages and Autoload	2-2
Regular Expressions.	2-2
Using Regular Expressions With Implicit Collections	2-3
Using Regular Expressions With Hierarchy.	2-3
Anchoring Regular Expressions	2-4
Using Regular Expressions With Buses	2-4
Extending Procedures	2-5
Default Procedure Attributes	2-5
Changing the Aspects of a Procedure.	2-5
Using the -define_args Option of the define_proc_attributes Command.	2-6
Parsing Arguments Passed in to a Tcl Procedure	2-8
TclPro Toolkit	2-9
Installation Requirements	2-10
Checking the Syntax and Semantics of Your Scripts.	2-10
Limitations Using the Synopsys Syntax Checker	2-11

Running the Synopsys Syntax Checker	2-11
Bytecode-Compiled Files	2-13
Debugging Scripts	2-13
Launching Prodebug From PrimeTime	2-14
TclPro Limitations	2-14

3. Using PrimeTime With Other Synopsys Tools

Using PrimeTime With Design Compiler	3-2
Features Specific to PrimeTime.	3-2
Timing Analysis Differences.	3-2
Paths.	3-2
Transition Time	3-3
current_design Command	3-3
Command Scripts	3-4
Sharing Design Compiler and PrimeTime Scripts	3-4
Synopsys Design Constraints Formatted Script Files.	3-5
Reading .db Files With Back-Annotated Data	3-5
Path Groups in Design Compiler	3-6
Manual Netlist Editing	3-6
Automatic Uniquifying of Blocks.	3-8
Resolving Library Cells	3-9
Estimating Delay Changes.	3-10
Customizing Estimation Columns	3-13
Sizing Cells	3-16
Inserting Buffers.	3-17
Swapping Cells	3-18
Renaming Cells or Nets.	3-19
User Function Class Support.	3-20
Using the fix_eco_drc Command	3-21
Writing Change Files	3-22
Choosing an Output Format	3-22
Specifying Buffer Insertion Cells and Net Names	3-22
Controlling Library Logical Names and Prepending File Names	3-23
Reading Changes Into Layout	3-24
Incremental Extraction With StarRC	3-24
Applying Netlist Changes.	3-25

Reading ECO Parasitics	3-25
Clearing Netlist Changes	3-26
Incremental Extraction Flow Examples	3-26
Writing Astro Change Files	3-27
4. Context Characterization	
Context Characterization Overview	4-2
Setting Synthesis or Optimization Constraints	4-2
Performing Subdesign Timing Analysis	4-2
Deriving the Context of a Subdesign	4-3
Clock Information	4-4
Input and Output Delay Times	4-4
Point-to-Point Timing Exceptions	4-4
Constant Logic Values on Inputs	4-5
Input Drive Strength and Port Capacitance	4-5
Wire Load Models	4-5
Design Rule Checks	4-6
Annotated Delays and Parasitics	4-6
Input Delay and Port Capacitance	4-8
Writing Physical Information	4-11
Reporting the Timing Context	4-12
Generating Scripts for Characterized Contexts	4-12
Removing Context Information	4-13
Limitations of Context Characterization	4-13
5. Advanced Analysis Techniques	
Time Borrowing in Latch-Based Designs	5-2
Borrowing Time From Logic Stages	5-2
Latch Timing Reports	5-3
Maximum Borrow Time Adjustments	5-7
Time Borrowed and Time Given	5-11
Limiting Time Borrowing	5-13
Path-Based Timing Analysis	5-14
Setting Recalculation Limits	5-18
Composite Current Source (CCS) Receiver Model for Path-Based Analysis	5-19

High Capacity Mode	5-19
Fast Analysis Mode	5-20
Parallel Arc Path Tracing	5-21
Support for Retain Arcs	5-22
True and False Path Detection	5-22
Reporting True or False Paths	5-23
Reporting True Paths	5-24
Justifying Paths	5-24
Finding the Longest True Path	5-26
Changing the Backtrack Limit	5-27
Changing the Prove-False Backtrack Limit	5-27
Using the True Delay Function	5-28
Long Path Example	5-29
Asynchronous Logic Analysis	5-31
Combinational Feedback Loop Breaking	5-33
Dynamic Loop Breaking	5-33
Specifying Loop Break Points	5-34
Unrelated Clocks	5-35
Three-State Bus Analysis	5-35
Limitations of the Checks	5-36
Disabling the Checks	5-36
Bus Contention	5-37
Floating Buses	5-37
Three-State Buffers	5-37
Performing Transient Bus Contention Checks	5-38
Performing Floating Bus Checks	5-39
Fast Multidrive Delay Analysis	5-40
Parallel Driver Reduction	5-41
Invoking Parallel Driver Reduction	5-43
Working With Reduced Drivers	5-43
Data-to-Data Checking	5-44
Data Check Examples	5-44
Data Checks and Clock Domains	5-47
Library-Based Data Checks	5-48

Data Propagation Through Generated Clocks	5-48
Interdependent Setup and Hold Pessimism Reduction	5-48
Use Model for SHPR	5-49
Setup-Preferred Slack Improvement.	5-49
Hold-Preferred Slack Improvement.	5-49
Total Negative Slack Improvements	5-50
SHPR Optimization Constraints	5-50
SHPR Optimization Mechanism	5-50
SHPR User Interface	5-51
SHPR Examples	5-52
Setup Preferred Slack Improvement Example	5-52
Total Slack Improvement Example	5-53
Liberty Format Extension.	5-54
6. Advanced On-Chip Variation (AOCV)	
Introduction	6-2
AOCV Flow	6-2
Graph-Based AOCV Solution	6-2
Path-Based AOCV Solution	6-3
Specifying the Scope of the AOCV Analysis.	6-4
Importing AOCV Information.	6-4
Specifying Derate Tables	6-5
File Format for AOCV	6-6
Specifying Random Coefficients	6-9
Guard-Banding in AOCV	6-10
Incremental Timing Derates.	6-10
Using OCV Derates in AOCV Analysis	6-12
7. Ideal Network Support	
Introduction to Ideal Networks	7-2
Propagating Ideal Network Properties	7-2
Using Ideal Networks	7-2
Using Ideal Latency	7-4
Using Ideal Transition	7-4

8. SDF Back-Annotation

Overview of SDF Back-Annotation	8-2
Reading SDF Files	8-2
Annotating Timing From a Subdesign Timing File	8-3
Annotating Load Delay	8-3
Annotating Timing Checks	8-3
Reading the File	8-4
Removing Annotated Timing Checks and Delays	8-5
Managing Large Files	8-5
Reporting Delay Back-Annotation Status	8-6
Reporting Annotated or Nonannotated Delays	8-6
Reporting Annotated or Nonannotated Timing Checks	8-7
Faster Timing Updates in SDF Flows	8-7
Annotating Conditional Delays From SDF	8-8
Writing an SDF File	8-10
SDF Constructs	8-10
SDF Delay Triplets	8-11
SDF Conditions and Edge Identifiers	8-11
Reducing SDF for Clock Mesh/Spine Networks	8-12
PORT Construct	8-13
Normalizing Multidriven Arcs for Simulation	8-14
Writing VITAL Compliant SDF Files	8-16
Removing Annotated Delays and Checks	8-17
Removing Annotated Delays	8-17
Removing Annotated Checks	8-18
Setting Annotations From the Command Line	8-18
Annotating Delays	8-19
Annotating Timing Checks	8-19
Annotating Transition Times	8-20
Generating Timing Constraints for Place and Route	8-20
Providing Constraint Coverage for the Entire Design	8-22

9. Parasitic Back-Annotation

Parasitic Data	9-2
--------------------------	-----

Lumped Parasitics	9-3
Setting Net Capacitance	9-3
Setting Net Resistance	9-3
Reduced and Detailed Parasitics	9-4
Annotating Reduced Parasitics	9-4
Annotating Detailed Parasitics	9-5
Supported File Formats for Parasitic Annotation	9-6
Characterization Trip Points	9-6
Reading Parasitics Files	9-9
Scaling Parasitic Values	9-11
Net-Specific Parasitic Scaling	9-11
Ground-Capacitance and Resistance Scaling	9-12
Coupling-Capacitance Scaling	9-12
Resetting Scale Parasitics	9-12
Reporting Scale Parasitics	9-13
Net-Specific Parasitic Scaling Examples	9-13
Incremental Timing Analysis	9-19
Incomplete Annotated Parasitics	9-20
Selecting a Wire Load Model for Incomplete Nets	9-20
Completing Missing Segments on the Net	9-21
Reporting Annotated Parasitics	9-22
Removing Annotated Parasitics	9-23
 10. Delay Calculation With Detailed Parasitics	
Overview of Delay Calculation	10-2
Nonlinear Delay Models (NLDM)	10-3
CCS Timing Models	10-5
Pin Capacitance Reporting	10-7
Guidelines to Address the CCS Extrapolation	
Warning Message (RC-011)	10-8
Guidelines for Characterizing Design Rule Constraints	10-8
Guidelines for Fixing RC-011 Warning Messages	10-9
Scaling With CCS Timing Libraries	10-9
Invoking Scaling	10-9
Guidelines for Scaling	10-10

Scaling Interpolation for Constraints	10-12
Scaling of Design Rule Constraints	10-12
Scaling Support for Multirail Cells	10-12

11. Low-Power Flow Support

Multivoltage Analysis	11-2
UPF Commands	11-3
Virtual Power Network	11-7
Setting Voltage and Temperature	11-8
Analysis With Multiple Voltages	11-10
Support for Fine-Grained Switch Cells in Power Analysis	11-11
Multivoltage Reporting and Checking	11-12
Collection (get_*) Commands	11-12
Reporting Commands	11-15
Using the check_timing Command	11-19
Voltage Set on Each Supply Net Segment	11-19
Supply Net Connected to Each PG Pin of Every Cell.	11-19
Compatible Driver-to-Load Signal Levels	11-19
Impact of Correlated Supplies on Signal Level Checking	11-22
UPF Supply Sets	11-23
Library PG Tcl File	11-24
Default Power and Ground Pin Names	11-25
Power Domain Mode of Release Z-2007.06	11-26
Multivoltage Method Prior to Release Z-2007.06	11-28
Setting Operating Conditions on Cells.	11-29
Setting Rail Voltages Directly on Cells.	11-30

12. Object Attributes

Using Attributes	12-2
Defining User Attributes	12-2
Importing User-Defined Attributes	12-3
Saving Design Attributes.	12-5
Attribute Names and Usage	12-5

Cell Object Class Attributes	12-6
Clock Object Class Attributes	12-14
Design Object Class Attributes	12-20
Library Object Class Attributes	12-30
Library Cell Object Class Attributes	12-32
Library Pin Object Class Attributes	12-35
Library Timing Arc Object Class Attributes	12-40
Net Object Class Attributes	12-42
Path Group Object Class Attributes	12-51
Pin Object Class Attributes	12-52
Port Object Class Attributes	12-71
Timing Arc Object Class Attributes	12-92
Timing Path Object Class Attributes	12-95
Timing Point Object Class Attributes	12-102
Using Paths to Generate Custom Reports	12-103
Using Arcs to Generate Custom Reports	12-105
Creating a Collection of Library Arcs	12-106
Reporting Library Data and Driver Information	12-107

Appendix A. Writing Mapped SDF Files

Specifying Timing Labels in the Library	A-2
Specifying the min_pulse_width Constraint	A-2
Accessing the min_pulse_width Constraint	A-3
Specifying the min_period Constraint on a Pin	A-3
Using SDF Mapping	A-3
SDF Mapping Notation	A-5
SDF Mapping Comments	A-5
SDF Mapping Variables	A-6
Supported SDF Mapping Functions	A-7
SDF Mapping File Syntax	A-9
SDF Mapping Assumptions	A-10
Bus Naming Conventions	A-10
Header Consistency Check for SDF Mapping Files	A-10
Labeling Bus Arcs	A-10

SDF Mapping Limitations	A-12
Mapped SDF File Examples	A-12
Library File for Cells EXMP and FF1	A-12
Mapped SDF File	A-16
Three-State Buffers	A-17

Index

Preface

This preface includes the following sections:

- [What's New in This Release](#)
- [About This User Guide](#)
- [Customer Support](#)

What's New in This Release

Information about new features, enhancements, and changes, along with known problems and limitations and resolved Synopsys Technical Action Requests (STARs), is available in the *PrimeTime Release Notes* in SolvNet.

To see the *PrimeTime Release Notes*,

1. Go to the Download Center on SolvNet located at the following address:

<https://solvnet.synopsys.com/DownloadCenter>

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.

2. Select PrimeTime, and then select a release in the list that appears.

About This User Guide

The *PrimeTime Advanced Timing Analysis User Guide* describes advanced topics related to performing static timing analysis using PrimeTime. It supplements the basic information provided in the *PrimeTime Fundamentals User Guide*.

Audience

This user guide is for design engineers who use PrimeTime for static timing analysis.

Related Publications

For additional information about PrimeTime, see the documentation on SolvNet at the following address:

<https://solvnet.synopsys.com/DocsOnWeb>

You might also want to refer to the documentation for the following related Synopsys products:

- PrimeTime PX, PrimeTime SI, and PrimeTime VX
- Design Compiler
- IC Compiler
- Library Compiler and Liberty NCX

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
Courier	Indicates syntax, such as <code>write_file</code> .
<i>Courier italic</i>	Indicates a user-defined value in syntax, such as <code>write_file design_list</code> .
Courier bold	Indicates user input—text you type verbatim—in examples, such as <code>prompt> write_file top</code>
[]	Denotes optional arguments in syntax, such as <code>write_file [-format fmt]</code>
...	Indicates that arguments can be repeated as many times as needed, such as <code>pin1 pin2 ... pinN</code>
	Indicates a choice among alternatives, such as <code>low medium high</code>
Control-c	Indicates a keyboard combination, such as holding down the Control key and pressing c.
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Edit > Copy	Indicates a path to a menu command, such as opening the Edit menu and choosing Copy.

Customer Support

Customer support is available through SolvNet online customer support and through contacting the Synopsys Technical Support Center.

Accessing SolvNet

SolvNet includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. SolvNet also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access SolvNet, go to the following address:

<https://solvnet.synopsys.com>

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to register with SolvNet.

If you need help using SolvNet, click HELP in the top-right menu bar.

Contacting the Synopsys Technical Support Center

If you have problems, questions, or suggestions, you can contact the Synopsys Technical Support Center in the following ways:

- Open a support case to your local support center online by signing in to SolvNet at <https://solvnet.synopsys.com>, clicking Support, and then clicking “Open A Support Case.”
- Send an e-mail message to your local support center.
 - E-mail support_center@synopsys.com from within North America.
 - Find other local support center e-mail addresses at <http://www.synopsys.com/Support/GlobalSupportCenters/Pages>
- Telephone your local support center.
 - Call (800) 245-8005 from within North America.
 - Find other local support center telephone numbers at <http://www.synopsys.com/Support/GlobalSupportCenters/Pages>

1

Overview of Advanced Topics

Prior to consulting this user guide, you should familiarize yourself with the *PrimeTime Fundamentals User Guide*, which covers basic topics, such as getting started, design data, clocks, analysis conditions, and timing reports. This user guide covers more advanced topics, such as tool command language (Tcl) features, using Design Compiler, context characterization, back-annotation, and attributes.

This chapter describes the content in each of the other chapters of this manual:

- [Topics Covered in This Manual](#)

Topics Covered in This Manual

This manual covers a variety of topics for advanced users of PrimeTime. You should consider the *PrimeTime Fundamentals User Guide* a prerequisite for using this manual.

[Chapter 2, “Tcl Advanced Features,”](#) describes advanced features of the Tcl scripting language not covered in the *PrimeTime Fundamentals User Guide*, including Tcl packages and autoload, regular expressions (using `-regexp` to perform pattern matching), extending procedures, and the TclPro Toolkit.

[Chapter 3, “Using PrimeTime With Other Synopsys Tools,”](#) explains how to use PrimeTime effectively with Design Compiler. It explains the differences between PrimeTime and Design Compiler with respect to command syntax and timing analysis behavior.

[Chapter 4, “Context Characterization,”](#) describes how to use the `characterize_context` command in PrimeTime to capture the timing context of subdesigns in the chip-level timing environment. This information can be used to set the timing constraints of a subdesign during synthesis or logic optimization in Design Compiler, or to perform timing analysis hierarchically in PrimeTime.

[Chapter 5, “Advanced Analysis Techniques,”](#) describes a variety of advanced analysis topics, including true and false path detection, time borrowing, asynchronous logic analysis, three-state bus analysis, parallel driver reduction, and data-to-data checking.

[Chapter 6, “Advanced On-Chip Variation \(AOCV\),”](#) describes the advanced on-chip variation (AOCV) technology. It provides information about the graph-based and path-based solution. It explains the flow and shows the reporting aspects for this technology.

[Chapter 7, “Ideal Network Support,”](#) describes the creation of ideal networks, on which no design rule checking (DRC) constraints are run. This allows designers ignore networks (large unoptimized networks with high fanout and capacitance) in prelayout and, instead, focus on violations arising from other sources. The use of ideal networks reduces runtime because PrimeTime uses 'ideal timing' rather than internally calculated timing.

[Chapter 8, “SDF Back-Annotation,”](#) explains how to back-annotate a design with delay information contained in a Standard Delay Format (SDF) file, and how to write an SDF file from the PrimeTime database. Back-annotating delay information from a layout tool produces a more accurate timing analysis than estimating delays from wire load models.

[Chapter 9, “Parasitic Back-Annotation,”](#) explains how to back-annotate a design with detailed parasitic information (parasitic capacitors and resistors) for more accurate timing analysis. It explains the types of parasitic data accepted and the procedures for back-annotating, reporting, and removing parasitic data. It also explains the PrimeTime algorithm for automatically completing parasitic information that is incomplete.

[Chapter 10, “Delay Calculation With Detailed Parasitics,”](#) explains how PrimeTime calculates delays in the presence of detailed parasitics. Note that you do not need this information to use parasitic back-annotation. It is only provided in case you want to check the accuracy of the delay calculations performed by PrimeTime.

[Chapter 11, “Low-Power Flow Support,”](#) describes how PrimeTime supports using IEEE 1801™ Unified Power Format (UPF) to specify low-power features of a design, such as multiple power supplies and power gating. It explains how PrimeTime reads UPF scripts and provides examples demonstrating the use of UPF commands and infrastructure.

[Chapter 12, “Object Attributes,”](#) describes the object attributes contained in the PrimeTime design database (for example, the `number_of_pins` attribute attached to each cell) and how to gather attribute information for generating a custom report.

[Appendix A, “Writing Mapped SDF Files,”](#) provides detailed information about controlling the format of the SDF file produced by the `write_sdf` command. It is a supplement to Chapter 7, “SDF Back-Annotation.”

2

Tcl Advanced Features

The PrimeTime command interface is based on the Tcl command language. Basic principles of Tcl usage are presented in the *PrimeTime Fundamentals User Guide*.

This chapter explains some advanced features of Tcl in the following sections:

- [Tcl Packages and Autoload](#)
- [Regular Expressions](#)
- [Extending Procedures](#)
- [TclPro Toolkit](#)

Tcl Packages and Autoload

PrimeTime supports the standard Tcl package and autoload facilities. However, the `load` command is not supported, so packages that require shared libraries cannot be used. PrimeTime is shipped with the standard implementations of the packages that are part of the Tcl distribution. For reference, you can find these packages below the root of the Synopsys installation in the `auxx/tcllib/lib/tcl8.4` directory.

Add a new Tcl package to PrimeTime either by installing the package into the Synopsys tree or by adding a new directory to the `auto_path` variable in the application startup script (`.synopsys_pt.setup`).

PrimeTime provides two default locations for loading packages into the application:

- For application-specific Tcl packages, `auxx/tcllib/primetime`
- For packages that work with all Tcl-based Synopsys tools, `auxx/tcllib/snps_tcl`

For example, if you have a Tcl package called `mycompanyPT` that contains PrimeTime reporting facilities used by mycompany, you would create a directory called `mycompanyPT` under the `auxx/tcllib/primetime` directory and then put the `pkgIndex.tcl` file and Tcl source files for my package into that directory. You can use the package in your PrimeTime scripts by using the following command:

```
package require mycompanyPT
```

For detailed information about Tcl packages, see the `package` command man page, Tcl reference books, or visit the following Web site:

<http://tcl.activestate.com/doc/>

Regular Expressions

Many commands that create explicit collections allow you to choose between simple wildcard patterns, using the `*` and `?` wildcard characters, or complete regular expressions, using the `-regexp` option. To inform PrimeTime to use regular expressions when matching patterns, you can use the `-regexp` option. For example,

```
pt_shell> set gc [get_cells -regexp {i(1|2)_.*}]
```

Using the `-regexp` option causes PrimeTime to view the patterns argument as a regular expression. In addition, the pattern matching operators in the filter expression (`=~` and `!~`) also use regular expressions. The following two commands, which illustrate basic regular expressions, are equivalent:

```
pt_shell> get_cells blk* -filter "ref_name =~ AN*"
```

```
pt_shell> get_cells -regexp {blk.*} -filter "ref_name =~ AN.*"
```

If the patterns are equivalent, the `-regexp` option works the same way as wildcards on the results of the `get_cells`, `get_pins`, and `get_nets` commands. Both search capabilities locate the same objects if equivalent patterns are provided. For example, the following two commands retrieve the same results:

```
get_cell -regexp U1.*
get_cell U1*
```

Similarly, the following two commands retrieve the same results:

```
get_cell -regexp U1.* -hier
get_cell U1* -hier
```

To perform a case-insensitive search in PrimeTime, the `-regexp` and `-nocase` options must be combined.

PrimeTime uses the same regular expression engine as Tcl. For more information about the supported regular expression language and the `regexp` command, see the man page for `regexp` or `re_syntax`, or consult books on the subject in the engineering section of your local bookstore or library.

Using Regular Expressions With Implicit Collections

Commands that generate implicit collections do not directly support regular expressions. However, because an element of an implicit collection can be an explicit collection, you can use regular expressions indirectly. For example,

```
pt_shell> report_timing -from [list i1/* [get_pins -regexp $expr]]
```

Using Regular Expressions With Hierarchy

Mixing `-regexp` and `-hierarchical` options is the same as mixing a wildcard pattern with `-hierarchical` searches. Hierarchical searches with regular expressions are always compared to the leaf name of the object in question.

Anchoring Regular Expressions

By default, PrimeTime automatically anchors regular expression patterns. For example, the pattern `blk.*` is considered the same as `^blk.*$`, which is usually the intended behavior.

If you want a less restricted matching style, like the Tcl `regexp` command, prefix and suffix the pattern with `.*` to unanchor the pattern (that is, not do an exact match). For example, to get any cells that contain `U1`:

```
pt_shell> get_cells -regexp {.*U1.*}
```

This matches `U1`, `U11`, `U1A`, `U1_23`, plus `ZU1`, `ZZU1`, `hello_U1`, and so forth.

Using Regular Expressions With Buses

When using a regular expression to match buses, you must be very careful because the bus characters (`[]`) are part of the command language. In addition, the usage varies slightly depending on whether the command argument is a string or a list.

For a command argument which is a string, the following example shows the correct form. The expression argument to `filter_collection` is a string.

```
pt_shell> filter_collection -regexp [get_ports *] {full_name =~  
a\[[0-1]\]}
```

This regular expression matches ports `a[0]` and `a[1]`. A single backslash (`\`) must precede the real square brackets.

For a command argument which is a list, the syntax depends on how you specify the list. Consider the following examples, which use the `get_ports` command. The “patterns” argument to `get_ports` is a list.

```
pt_shell> get_ports -regexp [list {a\[[0-1]\]}]  
pt_shell> get_ports -regexp {{a\[[0-1]\]}}
```

These two commands are essentially equivalent. Proper list forms require single backslash quoting (`\`), just like string arguments. It is recommended that you use a properly formatted list for a list argument, especially in this situation. However, when you pass a single string into the “patterns” argument, double backslash quoting (`\\`) is required. For example,

```
pt_shell> get_ports -regexp {a\\[[0-1]\\]}
```

The double backslash is required because the promotion of the string to a list consumes one of the backslashes.

Extending Procedures

Tcl enables you to write reusable procedures. Basic Tcl procedures have positional arguments, allow default values for arguments, and allow a varying number of arguments. PrimeTime provides a set of extensions, which allows you or a process developer to define help text and semantic information for the arguments of a procedure. This is called defining procedure attributes.

The procedure attributes that you can define include

- A one-line informational text string that is shown with simple help
- The command group in which the procedure exists
- Whether the procedure can be modified
- Whether the body of the procedure can be viewed
- Help text and semantic rules for each argument of the procedure
- Whether the procedure can be abbreviated

The `define_proc_attributes` command defines attributes for a procedure. Use this command to modify the attributes of an existing procedure. After you have defined procedure attributes, parse the procedure arguments to your procedure with the `parse_proc_arguments` command.

Default Procedure Attributes

When you create a procedure by using the `proc` command,

- It is placed in the Procedures command group
- It has no help text for its arguments
- On invocation, abbreviate the procedure name, subject to the value of the `sh_command_abbrev_mode` variable
- View the body of the procedure using `info body`
- Possible to modify the procedure and its attributes

Changing the Aspects of a Procedure

Use the `define_proc_attributes` command to define the attributes of a Tcl procedure.

The `define_proc_attributes` command enables you to:

- Define the help text and semantic rules for each argument.
- Make the help for a procedure look like the help for an application command.

For a list of options that you can use with the `define_proc_attributes` command, see the man page.

Using the `-define_args` Option of the `define_proc_attributes` Command

The `-define_args` option of the `define_proc_attributes` command defines each possible procedure argument for use with help text. The value for the `-define_args` option is a list of lists where each list element defines one argument. Each list element has the following format:

```
arg_name option_help value_help [data_type] [attributes]
```

arg_name

Specifies the name of the argument.

option_help

Specifies a short description of the argument to use with the procedure's `-help` option.

value_help

Specifies the argument name for a positional argument. Otherwise, specify a one-word description for the value of a command-line option. This parameter has no meaning for a Boolean option.

data_type

Specifies the data type of an argument, which can be a `string` (the default), `list`, `Boolean`, `int`, `float`, or `one_of_string`. This parameter is optional.

attributes

Specifies additional attributes for an argument. This parameter is optional. [Table 2-1](#) describes additional attributes that can be used.

Table 2-1 Additional Argument Attributes

Attribute	Description
<code>required</code>	Specifies a required argument. The default. If you use the <code>required</code> attribute, you cannot use the <code>optional</code> attribute.
<code>optional</code>	Specifies an optional argument. If you use the <code>optional</code> attribute, you cannot use the <code>required</code> attribute.

Table 2-1 Additional Argument Attributes (Continued)

Attribute	Description
value_help	When argument help is shown for a procedure, this attribute also shows the valid values for a <code>one_of_string</code> argument. For data types other than <code>one_of_string</code> , this attribute is ignored.
values	List of allowable values for a <code>one_of_string</code> argument. This attribute is required if the argument type is <code>one_of_string</code> . If you use values with other data types, an error is displayed.
merge_duplicates	When an option appears more than once in a command, the values are concatenated into a list of values. The <code>parse_proc_arguments</code> receives a single entry, which is a list. When <code>merge_duplicates</code> is not specified, the right most value for the option is used.
script	Identifies the argument as a Tcl script. This is used in the TclPro flow, so that tools such as the <code>snps_checker</code> recurse into these arguments. For more information, see “TclPro Toolkit” on page 2-9 .

There are two reasons for using the `define_parse_attributes` command: consistent formatted help and syntax and semantic checking. To complete syntax and semantic checking, you must use the `parse_proc_arguments` command. If you do not use the `parse_proc_arguments` command, the procedure cannot respond to `-help`. However, you can always use the `help procedure_name -verbose` command.

The following procedure adds two numbers and returns the sum. For demonstration purposes, some unused arguments are defined.

```
pt_shell> proc plus {a b} { return [expr $a + $b]}
pt_shell> define_proc_attributes plus -info "Add two numbers" \
    -define_args { \
        {a "first addend" a string required} \
        {b "second addend" b string required} \
        {"-verbose" "issue a message" "" boolean optional}}
pt_shell> help -verbose plus
Usage: plus      # Add two numbers
    [-verbose]    (issue a message)
    a             (first addend)
    b             (second addend)
pt_shell> plus 5 6
11
```

Parsing Arguments Passed in to a Tcl Procedure

The `parse_proc_arguments` command parses the arguments passed to a Tcl procedure that was defined with the `define_proc_attributes` command. The `parse_proc_arguments` command works only if you use one procedure argument: the Tcl keyword `args`, meaning any number of arguments. Then, you pass the arguments to the `parse_proc_arguments`, which does the semantic checking for you.

Use the `parse_proc_arguments` command to parse arguments passed into a Tcl procedure. The value passed into the Tcl procedure is automatically converted to a valid one-of-string value option. It is no longer the original user-supplied string. Using the `parse_proc_arguments` command within Tcl procedures enables support for argument validation and support of the `-help` option. Typically, `parse_proc_arguments` is the first command called within a procedure. You cannot use the `parse_proc_arguments` command outside of a procedure. For syntax information, see the man page.

When a procedure that uses the `parse_proc_arguments` command is invoked with the `-help` option, `parse_proc_arguments` prints help information (in the same style as using the `help -verbose` command) and then causes the calling procedure to return. Similarly, if any type of error exists with the arguments (missing required arguments, invalid value, and so forth), the `parse_proc_arguments` command returns a Tcl error and the calling procedure terminates and returns.

If you do not specify `-help` and the specified arguments are valid, the array variable `result_array` contains each of the argument values that you entered, subscripted with the argument name. The argument names are not the names of the arguments in the procedure definition; the argument names are the names of the arguments as defined using the `define_proc_attributes` command.

The following procedure shows how the `parse_proc_arguments` command typically is used. The procedure `argHandler` accepts an optional argument of each type supported by the `define_proc_attributes` command, then prints the options and values received.

```
proc argHandler {args} {
    parse_proc_arguments -args $args results
    foreach argname [array names results] {
        echo "  $argname = $results($argname)"
    }
}

define_proc_attributes argHandler \
    -info "Arguments processor" \
    -define_args {
        {-Oos "oos help" AnOos one_of_string
            {required value_help {values {a b}}}}
        {-Int "int help" AnInt int optional}
        {-Float "float help" AFloat float optional}
```

```
{-Bool "bool help" "" boolean optional}
{-String "string help" AString string optional}
{-List "list help" AList list optional}}
{-IDup "int dup" AIDup int {optional merge_duplicates}}}
```

Invoking the `argHandler` procedure with `-help` generates the following output:

```
pt_shell> argHandler -help
Usage: argHandler          # argument processor
      -Oos AnOos           (oos help: Values: a, b)
      [-Int AnInt]         (int help)
      [-Float AFloat]      (float help)
      [-Bool]              (bool help)
      [-String AString]    (string help)
      [-List AList]        (list help)
      [-IDup AIDup]        (Int dup)
```

Invoking the `argHandler` procedure with an invalid option generates the following output and causes a Tcl error:

```
pt_shell> argHandler -Int z
Error: value 'z' for option '-Int' not of type 'integer'
(CMD-009)
Error: Required argument '-Oos' was not found (CMD-007)
```

Invoking the `argHandler` procedure with valid arguments generates the following output:

```
pt_shell> argHandler -Int 6 -Oos a -IDup 2 -IDup 3
-Int = 6
-Oos = a
-IDup = 2 3
```

Note:

Only the arguments entered are shown in the output of the `argHandler` procedure. To test to see if a specific argument was entered, you would do the following in your procedure:

```
if {[info exists results(-Bool)]}{
    ...
}
```

TclPro Toolkit

TclPro is an open-source toolkit for Tcl programming. These are some of the tools that come with TclPro:

- TclPro Checker (procheck)
Static script syntax checker

- TclPro Compiler (procomp)
Stand-alone bytecode compiler
- TclPro Debugger (prodebug)
Tcl debugger

The Synopsys Syntax Checker (snps_checker), based on TclPro Checker, is included with the PrimeTime distribution. It is not necessary to download the TclPro tools to access syntax checking.

This section discusses how you can leverage these tools in the Synopsys environment. For more information about the TclPro tools, see the following Web address:

<http://tcl.sourceforge.net>

Installation Requirements

To use any TclPro tools other than the syntax checker, you must install TclPro on your system. After you have installed TclPro on your system, you must define the environment variable `SNPS_TCLPRO_HOME`, where the TclPro installation exists. For example, if you installed TclPro version 1.5 at `/u/tclpro1.5`, you must set the `SNPS_TCLPRO_HOME` variable to point at that directory. PrimeTime uses this variable as a base for launching some of the TclPro tools. In addition, other Synopsys applications use this variable to link to the TclPro tools.

Checking the Syntax and Semantics of Your Scripts

The Synopsys Syntax Checker, based on the TclPro Checker, helps you find syntax and semantic errors in your Tcl scripts. Everything that you need for syntax and semantic checking is shipped with PrimeTime. In this environment, the following items are checked:

- Unknown options
- Ambiguous option abbreviation
- Use of exclusive options
- Missing required options
- Validation of literal option values for numerical range (range, <=, >=)
- Validation of one-of-string (keyword) options
- Recursion into constructs that have script arguments, such as the `redirect` and `foreach_in_collection` commands
- Warning of duplicate options overriding previous values

Limitations Using the Synopsys Syntax Checker

The following limitations apply to the Synopsys Syntax Checker:

- Command abbreviation is not checked. Abbreviated commands show up as undefined procedures.
- Aliases created with the `alias` command are not expanded and show up as undefined procedures.
- A few checks done when the application is running might not be checked using the `snps_checker` environment. For example, some cases where one option requires another option are not checked.
- Script size is an issue with `snps_checker` and TclPro 1.3 and 1.5. Scripts up to a few thousand lines can be reasonably checked, but beyond that, CPU time becomes a factor. Do not try to check extremely large scripts using `snps_checker`.
- PrimeTime allows you to specify Verilog-style bus names on the command line without rigid quoting, for instance, `A[0]`. This format with indices from 0 to 255 is checked. Wildcards `*` and `%` are also checked. Other forms, including ranges as `A[15:0]` would show as undefined procedures, unless represented as `{A[15:0]}`.
- User-defined procedures enhanced with the `define_proc_attribute` command are not checked. Because such procedures are declared with the `args` argument, no semantic errors are reported.

Running the Synopsys Syntax Checker

There are two ways to run `snps_checker` in the Synopsys environment. You can launch `snps_checker` from PrimeTime, or run it stand-alone.

To run `snps_checker` stand-alone, use the wrapper scripts provided with the PrimeTime installation. Running `snps_checker` directly does not work. For each platform there is a script in the appropriate bin directory. For PrimeTime you can find the script in the `sparcOS5/syn/bin/ptprocheck`, `linux/syn/bin/ptprocheck`, and so on.

To launch `snps_checker` from PrimeTime, you need to load a package provided with the installation, which is loaded as follows:

```
package require snpsTclPro
```

This makes the `check_script` command available. Pass the name of the script you want to check to the `check_script` command.

The following example shows the script that is used in `snps_checker`:

```
create_clock [get_port CLK] -p  
create_clock [get_ports CLK] -p -12.2
```

```

sort_collection
set paths [get_timing_paths -nworst 10 -delay_type mx_fall
-r]
my_report -from [sort_collection \
[sort_collection $a b] {b c d} -x]
foreach_in_collection x $objects {
    query_objects $x
    report_timing -through $x -thr $y -from a -from b -to z >
r.out
}
all_fanout -from X -clock_tree
puts [pwd foo]

```

After running the script, the extensions shipped with your release are loaded. In the example output it shows the Synopsys extensions being loaded. Each line in the output shows where a syntax or semantic error was detected. The offending command is shown with a caret (^) below the character that begins the offending token. The following example shows the output from `snps_checker`:

```

% /synopsys/2001.08/sparcOS5/syn/bin/ptprocheck ex1.tcl
Synopsys Tcl Syntax Checker - Version 1.0

Loading snps_tcl.pcx...
Loading primetime.pcx...
scanning: /disk/scripts/ex1.tcl
checking: /disk/scripts/ex1.tcl
/disk/scripts/ex1.tcl:1 (warnUndefProc) undefined
procedure:
get_port
get_port CLK
^
/disk/scripts/ex1.tcl:1 (SnpsE-MisVal) Value not specified
for
'create_clock -period'
create_clock [get_port CLK] -p
^
/disk/scripts/ex1.tcl:2 (SnpsE-BadRange) Value -12.2 for
'create_clock -period' must be >= 0.000000
create_clock [get_ports CLK] -p -12.2
^
/disk/scripts/ex1.tcl:3 (SnpsE-MisReq) Missing required
positional options for sort_collection: collection criteria
sort_collection
^
/disk/scripts/ex1.tcl:4 (badKey) invalid keyword "mx_fall"
must
be: max min min_max max_rise max_fall min_rise min_fall
get_timing_paths -nworst 10 -delay_type mx_fall -r
^
/disk/scripts/ex1.tcl:4 (SnpsE-AmbOpt) Ambiguous option
'get_timing_paths -r'
get_timing_paths -nworst 10 -delay_type mx_fall -r

```



```

^
/disk/scripts/ex1.tcl:5 (warnUndefProc) undefined
procedure:
my_report
my_report -from [sort_collection \
^
/disk/scripts/ex1.tcl:5 (SnpsE-UnkOpt) Unknown option
'sort_collection -x'
sort_collection \
[sort_collection $a b] {b c d} -x
^
/disk/scripts/ex1.tcl:9 (SnpsW-DupOver) Duplicate option
'report_timing -from' overrides previous value
report_timing -through $x -thr $y -from a -from b -to z > r.out
^
/disk/scripts/ex1.tcl:11 (SnpsE-Excl) Can only specify one
of
these options for all_fanout: -from -clock_tree
all_fanout -from X -clock_tree
^
/disk/scripts/ex1.tcl:12 (numArgs) wrong # args
pwd foo
^

```

Bytecode-Compiled Files

Bytecode-compiled files are created using the TclPro Compiler, procomp. These files have the following advantages over ASCII scripts:

- Efficient to load
- Secure because the contents are not readable
- Body of compiled Tcl procedures is hidden

PrimeTime can load bytecode-compiled scripts. To load a bytecode-compiled file, you source it like you do other scripts. You do not need any files other than the application to load bytecode-compiled files.

Debugging Scripts

The TclPro debugger is the most complex tool in the package. The debugger is similar to most source code debuggers. You can step over lines in the script, step into procedures, set breakpoints, and so on. It is not stand-alone—it requires the application to be running while you debug the script.

Launching Prodebug From PrimeTime

Running the TclPro debugger is dependent on the `SNPS_TCLPRO_HOME` environment variable. After this variable is set, debug scripts using the `debug_script` command. For more information about the `SNPS_TCLPRO_HOME` variable, see [“Installation Requirements” on page 2-10](#).

To launch prodebug from PrimeTime, you need to load a package provided with the installation. The package is as follows:

```
package require snpsTclPro
```

This makes the `debug_script` command available. Pass the name of the script you want to debug to the `debug_script` command. This command launches prodebug, and connects PrimeTime to it. The script is instrumented, and then you are ready for debugging. You can make subsequent calls to instrument the script by sourcing the file with the `source` command or with the `debug_script` command. For more information, see the TclPro documentation shipped with the debugger or the `debug_script` man page.

TclPro Limitations

TclPro is not supported on RS/6000. However, bytecode-compiled files can be loaded into this application on all platforms, including RS/6000; and `snps_checker` is supported on all UNIX platforms.

3

Using PrimeTime With Other Synopsys Tools

PrimeTime works well with other Synopsys tools. This chapter describes the differences between various Synopsys tools, such as Design Compiler, and how to use PrimeTime with them.

This chapter contains the following sections:

- [Using PrimeTime With Design Compiler](#)
- [Manual Netlist Editing](#)
- [Using the fix_eco_drc Command](#)
- [Writing Change Files](#)
- [Incremental Extraction With StarRC](#)
- [Writing Astro Change Files](#)

Using PrimeTime With Design Compiler

PrimeTime works very well with the Design Compiler synthesis tool, although there are some differences in command syntax and behavior between the two tools. The following sections describe how to make the two tools work together efficiently:

- [Features Specific to PrimeTime](#)
- [Timing Analysis Differences](#)
- [Command Scripts](#)
- [Reading .db Files With Back-Annotated Data](#)
- [Path Groups in Design Compiler](#)

Features Specific to PrimeTime

Some features of PrimeTime are not supported by Design Compiler, such as internal clocks, certain features of extracted timing models, and mode analysis (timing models with operating modes).

Design Compiler does not support the creation of clocks on internal pins of a leaf cell. You must specify these clocks at input or output pins that are contained in the fanin or fanout of the internal pin.

Design Compiler cannot analyze and optimize designs with timing models that use mode analysis. You must manually disable the inactive timing arcs to analyze and optimize a module in a particular operating mode.

Timing Analysis Differences

The differences in timing analysis behavior between PrimeTime and Design Compiler are described in the following sections.

Paths

The following differences apply to paths.

Time Borrowing for Hold Checks

PrimeTime disables short-path borrowing by default. (Short-path borrowing is time borrowing for hold checks at level-sensitive latches.) Design Compiler always allows borrowing for short paths.

To simulate the Design Compiler behavior in PrimeTime, set the `timing_allow_short_path_borrowing` variable to `true`. For more information about path borrowing, see the *Synopsys Timing Constraints and Optimization User Guide*.

Segmenting Paths: Load-Dependent Delays

In Design Compiler, if a timing report originates from a segmented path's new startpoint, Design Compiler transfers the load-dependent portion of the source gate's delay to the output net delay. This method can provide a better synthesis result, but it is not appropriate for static timing analysis. PrimeTime does not add the load-dependent delay to the net.

Transition Time

The following differences apply to transition time.

Bidirectional Pins

PrimeTime maintains separate transition time information for the driver and load parts of bidirectional inout pins.

Design Compiler keeps the maximum of driver and load transition times.

Clock Pins

PrimeTime propagates transition times to clock pins and uses this information for computation of clock-to-Q delay and transition time, and calculation of setup and hold delay.

Design Compiler uses 0.0 for computation of clock-to-Q transition time and uses propagated transition for calculation of clock-to-Q delay for setup and hold delay.

PrimeTime is more pessimistic than Design Compiler for setup. The best solution for pre-layout is the `set_clock_transition` command, which specifies a fixed transition time on clock pins. Both PrimeTime and Design Compiler use 0.0 as input transition time on D of level-sensitive latches when calculating transition time for D-to-Q arcs.

current_design Command

Design Compiler requires that you change the current design used to specify information locally, such as wire load models:

```
dc_shell> current_design BOTTOM
dc_shell> set_wire_load_model -name small
dc_shell> current_design MID
dc_shell> set_wire_load_model -name medium
dc_shell> current_design TOP
dc_shell> set_wire_load_model -name large
```

In PrimeTime, you achieve this by using the `current_instance` command or by specifying hierarchical cell names. Do not use `current_design` for this purpose. For example,

```
pt_shell> current_design TOP
pt_shell> set_wire_load_model -name large
pt_shell> set_wire_load_model -name medium [get_cells U1]
pt_shell> set_wire_load_model -name small [get_cells U1/U2]
```

Command Scripts

You can use command scripts in PrimeTime. A command script is a sequence of `pt_shell` commands in a text file. The file can be in ASCII, gzip, or bytecode format. The `source` command executes scripts in PrimeTime.

Sharing Design Compiler and PrimeTime Scripts

You can share scripts when you use both Design Compiler and PrimeTime on the same design. The easiest way to do so is to use Synopsys Design Constraints (SDC), as described in [“Synopsys Design Constraints Formatted Script Files” on page 3-5](#). Another method is to keep scripts that contain shared commands separate from scripts that contain tool-specific commands, as demonstrated in the following examples.

Note:

You cannot convert `dc_shell` Tcl scripts to `pt_shell` scripts, but you can source some `dc_shell` Tcl scripts directly into `pt_shell`. An alternative method is to write shared constraint files in Tcl using the SDC format. For more information, see [“Synopsys Design Constraints Formatted Script Files” on page 3-5](#).

The following `dc_shell` script includes another script called `timing_assertions.scr`:

```
current_design MID
set_wire_load_model -name medium
current_design TOP
set_wire_load_model enclosed
set_wire_load_model -name large
set_max_area 3000
set_dont_touch u13
include timing_assertions.scr
compile
```

The following script is part of the `timing_assertions.scr` script, which contains commands that both Design Compiler and PrimeTime understand:

```
set_operating_conditions WCCOM
create_clock -period 10 CLK
set_input_delay ...
set_output_delay ...
```

To use the Design Compiler script in PrimeTime, use the timing assertions script and a script that contains PrimeTime commands. Some PrimeTime commands are identical to Design Compiler commands in syntax. However, as mentioned earlier, commands that are tool-specific should be specified differently.

In this example, the `pt_shell` script contains commands that are equivalent to the commands in the `dc_shell` script shown earlier:

```
set_wire_load_mode enclosed
set_wire_load_model -name medium {u2 u4}
set_wire_load_model -name large
source timing_assertions.scr
set_max_area 3000
set_dont_touch u13
```

Synopsys Design Constraints Formatted Script Files

SDC formatted script files are Tcl scripts that use a subset of the commands supported by PrimeTime and Design Compiler. There are limitations placed on some of the supported commands. For example, not all options are supported in some cases. Although SDC is a subset of Tcl, it is not a full-function scripting language. It is intended to support the specification of timing constraints across various electronic design automation tools. It is not intended for complex scripting.

Within SDC, there is only one design: the current design. You cannot change the current design with SDC.

SDC files are read into PrimeTime with the `read_sdc` command, and they are written from PrimeTime using the `write_sdc` command. For more information about these commands, see the man pages.

Checking the Syntax of the SDC File

Check the SDC file and verify that it is syntactically and semantically correct by using the `read_sdc -syntax_only` command. This validates the script without having any effect on the design. For more information about SDC, see the *Using the Synopsys Design Constraints Format Application Note*. To locate this application note, see Documentation on the Web, which is available through SolvNet at the following address:

<https://solvnet.synopsys.com/DocsOnWeb>

Reading .db Files With Back-Annotated Data

Improve the performance of reading netlist .db format files with back-annotation information into PrimeTime by using one of the following procedures.

Procedure 1

1. In `dc_shell`, save the .db format file with no back-annotation information and write the back-annotation information using the Standard Delay Format (SDF) file.
2. In `pt_shell`, read the .db format file, which is not back-annotated, and the generated SDF file. This is faster than reading the back-annotated .db format file directly in PrimeTime.

Procedure 2

1. In `dc_shell`, save the timing assertions set on the design with the `write_script` command. Make sure you remove commands that are not timing related.
2. In `pt_shell`, read the back-annotated .db format file, but use the `-netlist_only` option of the PrimeTime `read_db` command to instruct PrimeTime to ignore the back-annotation.
3. Apply the timing assertions to the design by sourcing the generated script. Make sure the script is converted from a Design Compiler script to a PrimeTime script.

Path Groups in Design Compiler

In Design Compiler, use path groups to modify the cost function for optimization. You can assign paths or endpoints to named groups. Each group has a weighting value that you can specify, which contributes to the maximum delay cost for the design.

PrimeTime checks each timing path in the design for maximum delay violations: it compares the actual path delay for rising and falling signals to a target path delay. The worst violation for a group is the path with the largest negative slack.

You can specify path groups in PrimeTime. When you request a path-timing report, PrimeTime lists a report for each path group. Export this information to Design Compiler by using the `write_script` or `write_context` command or by budgeting the design.

When you create an explicit path group, you can also assign a critical range for the path group. The critical range defines a margin of delay for the path group during optimization in Design Compiler. A nonzero value allows the optimization of near-critical paths if the worst violation is not increased.

Manual Netlist Editing

PrimeTime provides commands for manually editing the netlist. Netlist editing is performed to address timing issues without having to iterate through the implementation tool. You can also use netlist editing to implement logical netlist changes to estimate their timing impact. You might want to edit the netlist because of a timing violation caused by a large net

capacitance, which in turn is caused by a large net fanout. For addressing common timing problems, increase the size of the driver cell using the `size_cell` command, or insert a buffer in the net to increase its drive by using the `insert_buffer` command. These capabilities are useful, especially late in a design cycle when the logic functionality is frozen, and only small edits can be made.

For a list of editing commands, see [Table 3-1](#).

Table 3-1 *Netlist Editing Commands*

Object	Task	Command
Buffer	Inserting a buffer	<code>insert_buffer</code>
	Removing a buffer	<code>remove_buffer</code>
Cell	Changing the size (or drive strength) of a cell	<code>size_cell</code>
	Creating a cell	<code>create_cell</code>
	Renaming a cell	<code>rename_cell</code>
	Removing a cell	<code>remove_cell</code>
Net	Adding a new net to the design	<code>create_net</code>
	Connecting nets to pins in the design	<code>connect_net</code>
	Disconnecting nets from pins in the design	<code>disconnect_net</code>
	Renaming a net	<code>rename_net</code>
	Removing a net	<code>remove_net</code>

To perform netlist editing, all you need is a linked design. You can edit any hierarchical cell in the netlist.

Note:

You should not perform substantial edits to the design for the purposes of adding new logic, removing out-of-date logic, or fixing widespread functional errors. Such significant design changes should be performed in the implementation tool.

As netlist edits are performed, the changes are recorded in an internal change list. When the editing of the netlist is complete, you can see the impact of the changes by using the standard timing analysis and reports. If you want to preserve a record of the changes that you have made, use the `write_changes` command to save the changes to a disk file. The changes can be written as a sequence of PrimeTime netlist editing commands, a sequence of IC Compiler netlist editing commands, or a text description of the changes.

Automatic Uniquifying of Blocks

By editing a hierarchical block that is one instance of multiple instances in a design, you make that block unique. PrimeTime “uniquifies” that block for you automatically. For example, if the edited block is an instance of BLOCK, PrimeTime renames it BLOCK_0 or some similar name, avoiding any names already used in the design.

Making an instance of a design in PrimeTime unique is somewhat different in comparison to other Synopsys tools because a new design is not created at the time the block becomes unique. A placeholder for the design is created, similar to the placeholder that exists when a design is removed from memory by using the `link_design -remove_sub_designs` command.

List designs created by automatic uniquifying by using the `list_designs -all` command. These designs become real only if and when you link the design again.

When a block is edited, it is uniquified, along with all blocks above it, up to the first singly-instanced block. Messages are generated when uniquifying occurs. For example, if you use the `size_cell` command to change the size of cell `i1/low/n1`, it causes the cells to be uniquified:

```
pt_shell> size_cell i1/low/n1 class/NR4P
Uniquifying 'i1/low' (low) as 'low_0'.
Uniquifying 'i1' (inter) as 'inter_0'.
Sized 'i1/low/n1' with 'class/NR4P'
1
```

Blocks are also marked as having been edited when netlist editing is performed. As soon as you edit a block, its parent and each level of the hierarchy up to the top-level has also been edited. This information is available from a Boolean attribute, `is_edited`, that is available on a design and on hierarchical cells. After you use the `size_cell` command, the `is_edited` attribute is on `i1/low` block. For example,

```
pt_shell> get_attribute [get_cells i1/low] is_edited
true
```

Resolving Library Cells

Many of the netlist editing commands pass a library cell as an argument, which can be a library cell object or the name of a library cell. Obtain the former by using the `get_lib_cells` command. The latter can be either

- The full name of the library cell, such as, `class/AND2`, in which case there is no ambiguity about what to link the cell to, or
- The base name of the library cell, such as, `AND2`, in which case the library cell base name must be resolved to a library.

The process of resolving library cell base names to libraries is described in this section.

Library cell resolution for netlist editing commands is used primarily for netlist editing in distributed multi-scenario analysis, where you cannot use full names of library cells because each scenario might have a different set of libraries. However, you can also use this feature in nonmulti-scenario PrimeTime. For more information about distributed multi-scenario analysis, see the *PrimeTime Distributed Multi-Scenario Analysis User Guide*.

Resolve library cell base names from the cell's current library or from a specific library that you define using the `-current_library` or `-library` option, respectively.

Use the `-current_library` option with the `size_cell`, `get_alternative_lib_cells`, and `report_alternative_lib_cells` commands. This option instructs PrimeTime to resolve the specified library cell base name from the existing cell's currently linked library only. This option prevents PrimeTime from searching any other libraries.

The `size_cell`, `insert_buffer`, and `create_cell` commands offer the `-libraries` option, with which you can specify a list of libraries to resolve a specified library cell name. The libraries are searched in the order specified in the command. This option prevents PrimeTime from searching any other libraries. The `-libraries` option is consistent with the `-libraries` option of the `get_alternative_lib_cells` and `report_alternative_lib_cells` commands.

In the absence of these options, PrimeTime resolves library cell base names from libraries in the following order:

- Link library of the current cell for the `size_cell`, `get_alternative_lib_cells`, or `report_alternative_lib_cells` command
- Libraries specified by the `22` variable
- Libraries specified by the `link_path` variable

Use the `-base_names` option with the `get_alternative_lib_cells` command to cause the command to return the alternative library cells as a list of library cell base names (a string) rather than a collection.

Estimating Delay Changes

When performing ECO design modifications, fully understanding the consequences of the timing changes is important before you modify your design. Making the change and evaluating the resulting timing can cause you to incur the cost of an incremental timing update.

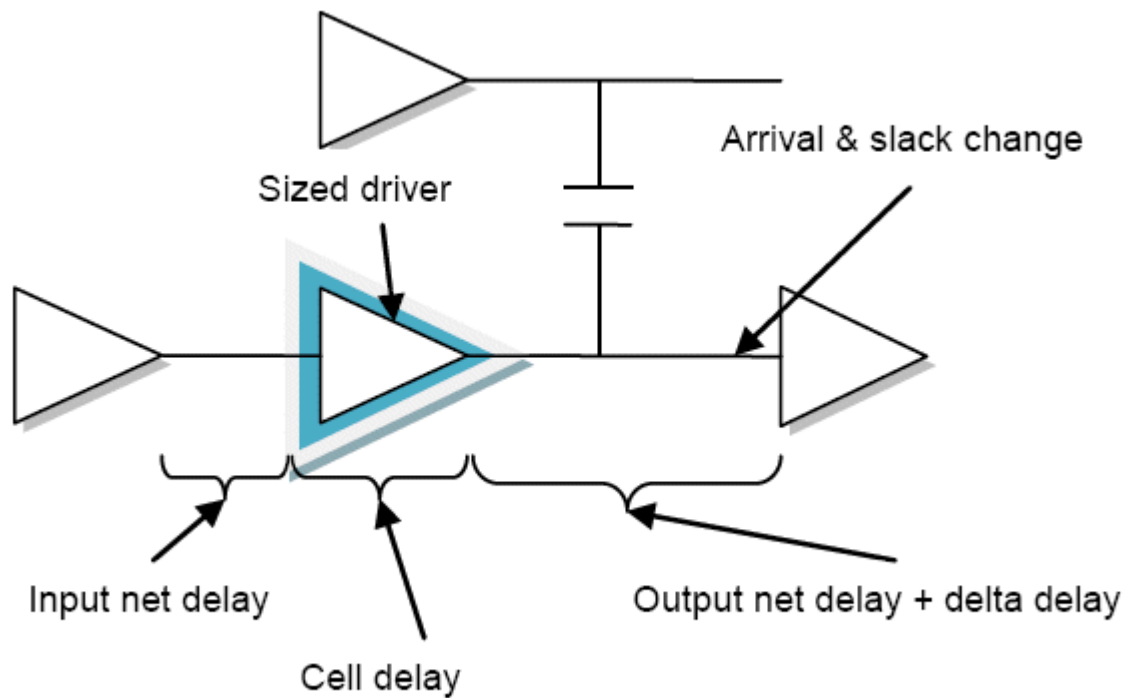
To improve the efficiency of the ECO flow, use the `estimate_eco` command. This command quickly computes and displays the ECO alternatives based on current timing values of a particular stage. The estimate takes into account the incoming transition times, output loading, fanout loading, detailed parasitics if present, and driver characteristics of the candidate cells.

The results are expected to reasonably predict, but not necessarily match, the actual timing resulting from subsequently performing that ECO command. Therefore, you should not rely on the `estimate_eco` command to get an accurate timing report because the results are not guaranteed. Rather, you should use the command to help choose the best design change among multiple options by showing the relative timing impact of each option. In particular, this command allows you to write efficient ECO scripts that can quickly choose optimal ECOs without relying on an expensive change, update, check, or modify inner loop.

Two types of ECO fixing methods that are available with the `estimate_eco` command are sizing and buffer insertion. Both of these methods change the netlist and, as a result, change the delay values of the associated stages. For example, sizing a victim driver cell

changes its crosstalk delay because of the driver resistance change. Moreover, it changes the previous stage delay and current stage delay due to input capacitance and driver resistance changes as shown in [Figure 3-1](#).

Figure 3-1 Timing Breakdowns



When using the `estimate_eco` command, you should use the following flow:

1. Show the available options.

In this stage, you evaluate possible ECO options. For example, you might want to size up a driver cell, `buf3`, in a path that has a setup timing violation. To find out the effects of substituting different drivers, you would use the following command that would generate the sample output, which also follows:

```
pt_shell> estimate_eco -type size_cell -max -rise
```

delay type : max_rise					
lib cell	area	stage_delay	arrival	slack	trans
mylib90/CKNXD12	19.05	0.0462	0.0462	0.1474	0.012
mylib90/INVD24	21.17	0.0265	0.0265	0.1671	0.030
mylib90/INVD20	18.35	0.0291	0.0291	0.1645	0.022
mylib90/CKNXD8	14.11	0.0610	0.0610	0.1326	0.105
mylib90/INVD3	3.53	0.1295	0.1295	0.0212	0.199
*mylib90/INVD2	2.82	0.1936	0.1936	-0.0013	0.292
mylib90/CKNXD1	2.82	0.4271	0.4271	-0.3214	0.326

Note:

This report shows all possible alternative library cells and associated timings with the asterisk (*) identifying the current cell. The rise or fall timing represents the worst timing through the cell output pins in the corresponding direction. The `stage_delay` column shows the delay of the inserted buffer or resized cell and the driven net. The arrival and slack columns represent the arrival time and slack at the load pins of the newly inserted buffer or resized cell. For more information about customizing the report output, see [“Customizing Estimation Columns” on page 3-13](#).

2. Estimate the stage delay improvements.

After deciding on the approaches for fixing the timing violation, you analyze the delay changes at the stage in which the netlist change is to occur. For example, you might want to insert a buffer to the input pin, `buf6/l`, and estimate how much the stage delay would change. The command syntax and sample output is as follows:

```
pt_shell> estimate_eco -type insert_buffer -inverter_pair \
             -max -rise -verbose \
             -lib_cell INVD
```

```
cell name : buf3
current lib cell: mylib90/INVD2
buffer lib cell : mylib90/INVD2
```

max_rise	current	estimate

input net delay	0.102954	0.052354
stage delay	0.000240	0.000120
cell delay	0.371027	0.175534

output net delay	0.206952	0.110121
buffer delay	0.023532	0.013895
delta delay	0.140543	0.051518
arrival time	0.371027	0.175534
slack	0.304100	0.499593
transition		
cell input	0.202001	0.101001
cell output	0.252001	0.151001
load input	0.261001	0.161001

3. Commit changes and verify.

If you are sure about the fix, run `size_cell`, `insert_buffer`, or both and then run the `update_timing` command to see real changes. Since the `estimate_eco` command shows only estimated delays, based on current-stage information, the real delay values after the change can be different due to the interaction between previous, next, and coupled stages.

Note:

The timing update in this process cannot always guarantee real silicon delays because PrimeTime does not have place and route information.

For more information about the `estimate_eco` command, including recommended methods for using the timing estimation data in scripts, see the man page. For more information about customizing the report output, see [“Customizing Estimation Columns”](#).

Customizing Estimation Columns

You can customize the format of the default compact output of the `estimate_eco` command. Use the `eco_estimation_output_columns` variable to specify the desired output columns for the `estimate_eco` command. The following column keywords are available:

- `area` - Shows the area of the alternative library cell.
- `stage_delay` - Provides stage delay of the alternative library cell.
- `arrival` - Provides arrival time at the most slack-critical load pin of the stage.
- `slack` - Shows slack through the cell.
- `transition` - Displays transition time at the most slack-critical load pin of the stage.
- `min_transition` - Provides the minimum transition DRC slack of pins surrounding the stage.
- `max_transition` - Provides the maximum transition DRC slack of pins surrounding the stage.
- `min_capacitance` - Shows the minimum capacitance DRC slack of pins surrounding the stage.

- `max_capacitance` - Displays the maximum capacitance DRC slack of pins surrounding the stage.
- `max_fanout` - Provides the worst maximum fanout DRC for any pin in the estimated stage.

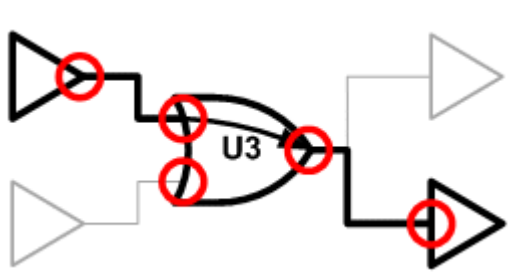
The default columns are `area`, `stage_delay`, `arrival`, and `slack`. By setting the desired columns with the `eco_estimation_output_columns` variable, you can specify the format once without requiring additional options for each use of the command.

If you specify the `size_cell` estimation type in the `estimate_eco` command, the design rule checking estimations include

- All pins of the cell being resized
- Previous driver pin
- Slack-critical load pin

For example, [Figure 3-2](#) shows the input and output pins with the `estimate_eco -type size_cell U3` command.

Figure 3-2 estimate_eco -type size_cell Example

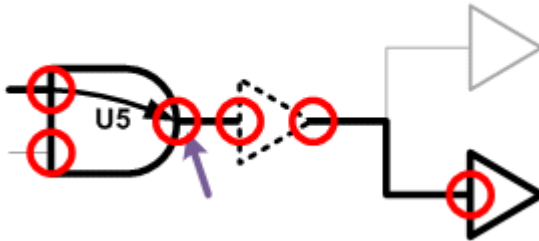


If you specify the `insert_buffer` estimation type in the `estimate_eco` command, the design rule checking estimations include

- All pins of the buffer being inserted
- All pins of the driver cell
- Slack-critical load pin

For example, [Figure 3-3](#) shows the input and output pins with the `estimate_eco -type insert_buffer U5/Z` command.

Figure 3-3 estimate_eco -type insert_buffer Example



The `eco_estimation_output_columns` variable also controls which sections appear in the verbose section. The following is an example of the output when you use the `-verbose` option of the `estimate_eco` command with the `max_transition` value included in the column list:

```
pt_shell> estimate_eco U2319 -lib_cell OR2X1 -v -sig 4
```

cell name : U2319		
current lib cell: OR2X2		
new lib cell: OR2X1		
max_rise	current	estimate

transition		
cell input	0.0941	0.0877
cell output	0.0800	0.1154
load input	0.0800	0.1154
max transition slack		
input net driver	4.4059	4.4123
cell input	4.4059	4.4123
cell output	4.4200	4.3846
load input	4.4200	4.3846
new lib cell: OR2X1		
max_fall	current	estimate

transition		
cell input	0.1217	0.1170
cell output	0.0821	0.1081
load input	0.0821	0.1081
max transition slack		
input net driver	4.3783	4.3830
cell input	4.3783	4.3830
cell output	4.4179	4.3919
load input	4.4179	4.3919

Sizing Cells

Replace the library cell referenced by a leaf cell in the netlist with a new cell by using the `size_cell` command. The result is a cell with new characteristics with respect to delay calculation, which results in different timing.

When you use the `size_cell` command and the targeted cell is not functionally equivalent, an error message appears. For example,

```
pt_shell> size_cell o_reg1 class/NR4P
Error: Could not size 'o_reg1' ('class/FD2') with 'class/
NR4P':
Cells not functionally equivalent. (NED-005)
Error: No changes made. (NED-040)
0
```

Use the `get_alternative_lib_cells` command to find a compatible library cell.

```
pt_shell> get_alternative_lib_cells o_reg1
{"class/FD2P"}
```

For example, to size the cell `o_reg1 class/FD2P`, enter

```
pt_shell> size_cell o_reg1 class/FD2P
Sized 'o_reg1' with 'class/FD2P'
1
```

A library might contain several alternative library cells. For example, `get_alternative_lib_cells` might return this:

```
pt_shell> get_alternative_lib_cells o_reg1
{"class/FD2P1", "class/FD2P2", "class/FD2P3"}
```

Which cell to use for replacement might not be obvious. Use the `report_alternative_lib_cells` command to obtain the slack at the outputs of the leaf cell. For example, you can use the `report_alternative_lib_cells` command for a report, such as this:

```
pt_shell> report_alternative_lib_cells o_reg1

*****
Report : alternative_lib_cells o_reg1 -delay_type max
Design : counter/0 (unknown)
Version: 2000.11
Date   : Wed Nov 15 17:53:16 2000
*****
Alternative                               Slack
Library Cells
-----
```

```

class/FD2P3                1.88 (r)
class/FD2  *                -1.02 (r)
class/FD2P1                -2.88 (r)
class/FD2P2                -2.98 (r)

```

The report indicates that the cell class/FD2P3 would yield a design that produces positive slack at the output of the leaf cell. In this case, you would use the class/FD2P3 to size the leaf cell. To verify the state of the design, you can use the `report_timing` command.

Inserting Buffers

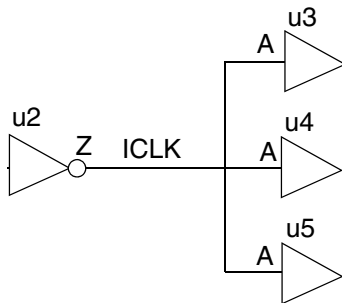
Use the `insert_buffer` command to add a buffer at one or more pins in the netlist. To remove a buffer from the netlist, you can use the `remove_buffer` command. For more information, see the `insert_buffer` and `remove_buffer` man pages.

Note:

If you insert a buffer when parasitics are present, the parasitics may not be preserved. See the `insert_buffer` man page for more information.

In the circuit shown in [Figure 3-4](#), pin u2/Z drives net ICLK, and ICLK has three loads: u3/A, u4/A, and u5/A.

Figure 3-4 Inserting Buffer

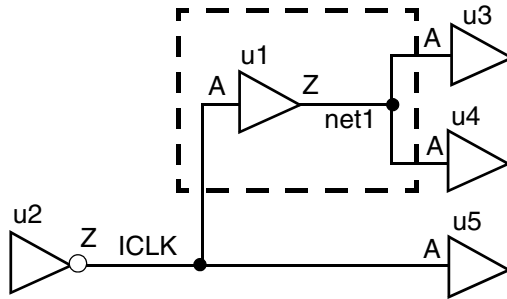


To buffer u3/A and u4/A, enter,

```
pt_shell> insert_buffer {u3/A u4/A} class/B1I
```

The `insert_buffer` command creates a new cell, u1, and a new net, net1. The input of the new buffer is always the existing net, in this case, ICLK. The output of the new buffer is always the new net, in this case, net1. Figure 3-5 shows the new circuit.

Figure 3-5 New Circuit



To change the prefix used to name the newly inserted buffers, you can use the `eco_instance_name_prefix` variable. By default it is set to "U."

Swapping Cells

The `swap_cell` command replaces the cells in the cell list with the design or library cell you specify in the `new_design` option.

Before the swap, the `swap_cell` command also checks whether the pinout of the cell you are swapping out and the pinout of the design or library cell you are swapping in are the same. For every pin of the cell you are swapping out, there must be a pin with the same name in the cell you are swapping in.

The `swap_cell` command always relinks the part of the design that has been replaced. Do not use the `link_design` command after you have used a `swap_cell` command; doing so often undoes the work of the `swap_cell` command. The PrimeTime data model is instance based. When you use the `swap_cell` command, an instance is modified. For example, U2/U0 is an instance of design x, and you swap U2/U0 for a different design. PrimeTime did not modify the design x; it modified the instance U2/U0. Therefore, an initial link reverts to the old design.

PrimeTime does not save information about cells that were swapped; that information is maintained only until the next link. However, the change list for the design records the fact that the swap occurred, and you can export this information using the `write_changes` command.

By default, the `swap_cell` command restores the constraints that were on the design before the swap occurred. If you are doing multiple swaps, saving the constraints using the `write_script` command is far more efficient and then you should use the `swap_cell` command with the `-dont_preserve_constraints` option.

When you have completed the swaps, restore the constraints by sourcing your script that you had written with the `write_script` command. If you are swapping one library cell for another, consider using the `size_cell` command, which is much more efficient. For example, to replace cells `u1`, `u2`, and `u3` in `TOP` with the `A` design (in `A.db`), enter

```
pt_shell> read_db A.db
pt_shell> current_design TOP
pt_shell> swap_cell {u1 u2 u3} A.db:A
```

To replace cells `u1`, `u2`, and `u3` in `TOP` with an `LC3` lib_cell in the `misc_cmos` library, enter

```
pt_shell> swap_cell {u1 u2 u3} [get_lib_cell \
misc_cmos/LC3]
```

Note:

The `swap_cell` command is intended to swap complex cells or hierarchical cells. When cells are swapped, a full timing update is incurred. You should not use the `swap_cell` command for simple cell resizing.

Renaming Cells or Nets

The `rename_cell` and `rename_net` commands allow you to change the name of a cell or net. When renaming a cell or a net using hierarchical names, make certain that the old and new names are at the same level of hierarchy. You can refer to a cell or net by its full hierarchical name as long as it is below the current instance.

To successfully rename a cell or net, you must meet the following conditions:

- PrimeTime must be able to find the cell or net.
- No more than one cell or net matches the cell or net pattern that you specify.
- No leaf cell or net is found to match the new name that you specify.
- A new name must be at the same hierarchical level as the cell or net that you are renaming.

The following example successfully renames cellA to cellB in the current instance, middle.

```
pt_shell> rename_cell cellA cellB
Information: Renamed cell 'cellA' to 'cellB' in design
'middle'.
(NED-008)
1
```

The following example renames a net at a level below the current instance. In this example, u1 and u1/low are instances of designs that have multiple instances; therefore, they are uniquified as part of the editing process.

```
pt_shell> rename_net [get_net u1/low/w1] u1/low/netA
Uniquifying 'u1/low' (low) as 'low_0'.
Uniquifying 'u1' (inter) as 'inter_0'.
Information: Renamed net 'w1' to 'netA' in 'middle/u1/low'.
(NED-009)
1
```

PrimeTime saves the name changes using the `rename_cell` and `rename_net` commands for output with the `write_changes` command. For additional information, see the man page for the `rename_cell` or `rename_net` command.

User Function Class Support

For some complex library cells, the `user_function_class` attribute is required to describe the functional behavior of the cell instead of the `function_id` attribute. With support for the `user_function_class` library cell attribute, what-if analysis cell resizing is possible for these complex cells.

Although PrimeTime scales cells with the same `function_id` attribute, you are also able to define their own function class. For example,

```
data buffers
clock buffers
delay buffers (used to fix hold times)
```

The foundry could provide these function classes to ensure that the proper cells are chosen for upsizing. You can set the `user_function_class` attribute using the `set_user_attribute` command, in Design Compiler, or in the source `.lib` file. For more information about setting this attribute, see the Design Compiler documentation.

Using the `fix_eco_drc` Command

The `fix_eco_drc` command automatically generates `size_cell` and `insert_buffer` commands to fix or reduce maximum transition, maximum capacitance, and maximum fanout design rule violations, while minimizing the impacts on timing and area. Although design rule checking fixing takes priority over timing, the fixes are chosen to minimize the impact on timing. The behavior of this command is similar to that of the `fix_eco_timing` command except that it fixes design rule violations instead of timing violations.

To fix only maximum transition design rule violations using only cell sizing and to generate a verbose fixing report, use the following command:

```
pt_shell> fix_eco_drc -type max_transition -methods {size_cell} -verbose
```

To fix only maximum transition design rule violations using only buffer insertion with buffers BUFX1, BUFX2, and BUFX3 and to generate a verbose fixing report, use the following command:

```
pt_shell> fix_eco_drc -type max_transition -methods {insert_buffer} \
    -buffer_list {BUFX1 BUFX2 BUFX3} -verbose
```

The fixing report shows the number of remaining DRC violations not fixed and, of these, the number that cannot be fixed and the number that might be fixed by running the `fix_eco_drc` command again. For example,

```
Remaining Violations:
Violation Type                               Count
-----
Total remaining violations                    2
Unfixable violations                         1
```

After the design has been fixed, use the `write_changes` command to write the `size_cell` and `insert_buffer` commands used in the fixing process into a script file. The file can then be used in a physical implementation tool such as IC Compiler to implement the changes.

For more information about using the `fix_eco_drc` command, see the man page for the command.

Writing Change Files

After the desired netlist edits have been performed, whether through manual editing, automated editing, or a combination of the two, you can use the `write_changes` command to generate an ECO change file. The `write_changes` command can write the design changes in multiple output formats, and there are options available to control how the design change information is written. The following sections provide more information.

- [Choosing an Output Format](#)
- [Specifying Buffer Insertion Cells and Net Names](#)
- [Controlling Library Logical Names and Prepending File Names](#)
- [Reading Changes Into Layout](#)

Choosing an Output Format

The `write_changes` command provides a `-format` option that allows changes to be written out in multiple formats. These formats are:

- `ptsh` - Changes are written as Tcl script that can be sourced in PrimeTime.
- `icctcl` - Changes are written as Tcl script that can be sourced in IC Compiler or Design Compiler.
- `text` - Changes are written as human-readable text that can be parsed by external scripts for conversion into third-party formats.
- `dctcl` - Changes are written in the same format as the `icctcl` format. This format is deprecated in favor of other formats listed in this section.

Specifying Buffer Insertion Cells and Net Names

The `insert_buffer` command adds a buffer at one or more specified pins or ports. This command allows you the option of providing the names for new nets and cells added during buffer insertion. Multiple methods are available to control the new cell and net names. You can choose one of the following methods:

- Specify no naming information, and choose to use the default naming rules.
- Use the `eco_instance_name_prefix` and `eco_instance_name_prefix` variables to change the default naming rules.
- Explicitly specify the `-new_cell_names` and `-new_net_names` options of each manually-executed `insert_buffer` command.

To ensure that the buffer insertions of PrimeTime are reproduced correctly, the output from the `write_changes` command in the `icctcl` and `ptsh` formats always specifies the new cell and net names for any `insert_buffer` operations, even if you did not explicitly specify them. This ensures a robust interface between PrimeTime and IC Compiler and also maintains a consistent netlist between the netlist in static timing analysis and the netlist in the implementation tools.

Controlling Library Logical Names and Prepending File Names

When the `create_cell`, `size_cell`, and `insert_buffer` operations are written to the change file, the commands must reference library cells. You can control the formatting of these library cell references by using the variables described below. You should set these variables before netlist changes, as they control how PrimeTime records the changes in memory. Because the variables control how the changes are stored in the change list, they affect all of the `write_changes` output formats.

When the `eco_write_changes_prepend_libname_to_libcell` variable is set to `true`, the logical library name is included in the library cell definitions. When this variable is set to its default of `false`, the file contains only the reference cell name without the logical library name. Ensure this variable is set to `false` if the logical library names used in the implementation tool differ from those that are used in the PrimeTime analysis.

[Example 3-1](#) and [Example 3-2](#) compare the output Tcl file for both settings:

Example 3-1 Change File, Prepending of Logical Library Name Set to true

```
insert_buffer -new_cell_names ECO_1 \
-new_net_names net_1 [get_pins {I7/A}] lib1/buf41
```

Example 3-2 Change File, Prepending of Logical Library Name Set to false

```
insert_buffer -new_cell_names ECO_1 \
-new_net_names net_1 [get_pins {I7/A}] buf41
```

When the library name is prepended by having the `eco_write_changes_prepend_libname_to_libcell` variable set to `true`, an additional control becomes available. In addition, if you want to prepend the exact library file names to the logical names in the change list, you can also set the `eco_write_changes_prepend_libfile_to_libcell` variable to `true`. By default this variable is set to `false`.

[Example 3-3](#) shows the `write_changes icctcl` output with `libfile` prepending capabilities enabled:

Example 3-3 Change File, Prepending of Library File Name Set to false

```
insert_buffer -new_cell_names ECO_1 -new_net_names net_1 [get_pins {I7/A}] \
lib.db:lib1/buf41
```

This feature is typically used only when multiple libraries with the same logical file name are present in the tool reading the change file, and differentiate between these libraries is important. In most cases, the default value of `false` is suitable.

Reading Changes Into Layout

After the change file is written, the changes can be read into the layout tool. For the postroute flow using IC Compiler, you can push the design changes into the routed layout database by performing the following tasks:

1. Generate the list of changes for IC Compiler by executing the following command:

```
write_changes -format icctcl
```
2. In IC Compiler, source the change list ECO file.
3. Execute the `legalize_placement` and `route_zrt_eco` IC Compiler commands.
4. Run StarRC extraction.
5. After implementation, run PrimeTime to validate the QoR.
6. Repeat additional ECO fixing iterations through PrimeTime and layout, as necessary.

Incremental Extraction With StarRC

After the changes have been written from PrimeTime, two flows through layout are available. Most commonly, a new full-design netlist is generated from layout, and a new full-design parasitics file is generated that matches the new netlist. These new netlist and parasitics files are read into a new PrimeTime analysis session.

An incremental extraction flow with StarRC is also available. This flow can improve productivity in large designs by retaining the netlist edits already performed in PrimeTime, and reading in only the parasitics that have changed due to buffer insertions, cell insertions, cell legalization, or net rerouting. This results in a reduced Standard Parasitic Exchange Format (SPEF) or Synopsys Binary Parasitic Format (SBPF) file. You can then read the ECO file into the previous PrimeTime or PrimeTime SI analysis by using the `read_parasitics -eco` command.

The following sections explain the steps in the incremental extraction flow in more detail:

- [Applying Netlist Changes](#)
- [Reading ECO Parasitics](#)
- [Clearing Netlist Changes](#)

- [Incremental Extraction Flow Examples](#)

Applying Netlist Changes

To enable PrimeTime to read incremental parasitics information from StarRC, you must issue the following command before the design is initially read in:

```
set_program_options -enable_eco
```

This places PrimeTime into a special ECO mode where it retains additional information that allows it to stitch a new incremental parasitics file onto the existing parasitics. At this point, any needed netlist editing operations can be performed.

After the desired netlist edits have been completed, you can use the `write_changes` command to write a change file for IC Compiler. After the changes have been applied to the design in IC Compiler, incremental extraction in StarRC is performed. For the incremental flow to work, you must not apply any other structural netlist changes, such as cell creation or removal or net connectivity, in IC Compiler, except for those changes written out by PrimeTime. Cell movement and net rerouting are acceptable, as these changes are brought back through incremental extraction.

Reading ECO Parasitics

The `read_parasitics -eco` option allows you to read in an incremental parasitics file from StarRC. For more information about the `read_parasitics` command, see the man page.

Using the `-eco` option can shorten the overall ECO cycle time. All the other options, such as `-path` or `-increment`, should be exactly the same as before.

If the number of changes is small, reading in the partial parasitics file triggers only an incremental timing update. This results in faster turnaround times when you are checking the feasibility of various ECO changes. After performing initial analysis in PrimeTime, you can either leave your `pt_shell` running or save a session and restore it later to consume ECO parasitics.

In ECO mode, PrimeTime remembers the original parasitics files that were read in. When incremental parasitics are read in, PrimeTime attempts to determine which original parasitics file corresponds to the incremental parasitics file. This association is needed so that coupling between the incrementally extracted nets and the original nets can be properly restored. Normally, PrimeTime is able to determine the location of the original parasitics file. If this is not possible, you can use the `read_parasitics -original_file_name` option to specify the original parasitics file to which the ECO file corresponds.

Note:

You cannot use the parasitics files written by PrimeTime for incremental extraction flows if the source of the parasitics comes from multiple files. For example, if your design is annotated with multiple parasitics files and you use the `write_parasitics` command to convert the in-memory parasitics to a single parasitics file, you cannot use the resulting single parasitics file for ECO flows.

Clearing Netlist Changes

After the incremental parasitics have been read into PrimeTime, the internal change list should be erased for the next round of editing. This ensures that the next `write_changes` command only writes the new changes going forward, rather than including the previous changes that have already been incorporated into the layout.

The `write_changes -reset` option clears any netlist change lists. When you use this option, PrimeTime no longer retains any history of previous netlist editing commands. This only clears the list of changes remembered by the `write_changes` command. It does not actually undo any of the design changes that have been made.

Incremental Extraction Flow Examples

This section shows an example of a typical incremental extraction ECO flow. The first section shows the initial analysis performed in ECO mode. In this example, the `fix_eco_timing` command is used to fix some timing violations:

```
set_program_mode -enable_eco
read_verilog ...
read_sdc ...
read_parasitics my_spef.spef -keep_capacitive_coupling
update_timing
set eco_instance_name_prefix {UECO}
fix_eco_timing -type setup
fix_eco_timing -type hold
update_timing
report_timing
write_changes -format icctcl -output eco.tcl
save_session my_session
```

At this point, the netlist edits have already been performed in the current PrimeTime session. You can save this PrimeTime session to a disk using the `save_session` command.

Next, you can take the change file into IC Compiler and update the layout. You then perform incremental extraction in StarRC. After you are finished with those tools, you return to PrimeTime. You can restore the session if the original analysis is not still running. You then use the following commands to apply the incremental parasitics back into the PrimeTime analysis:

```
pt_shell> restore_session my_session
pt_shell> read_parasitics -eco incr.spef -keep_capacitive_coupling
pt_shell> write_changes -reset
pt_shell> update_timing
pt_shell> report_timing
```

Writing Astro Change Files

When performing what-if analysis in PrimeTime and PrimeTime SI to investigate possible fixes for timing or system integrity problems, you can write out the netlist changes directly in native Astro formats. The `write_astro_changes` command writes out the structural netlist ECO operations and coupling separation directives. This is a more direct approach than the Verilog netlist method because it does not require importing a complete Verilog description into the design. Use the `write_astro_changes` command to generate an ECO change file.

Use the `write_astro_changes` command to write changes. If the `heco` format is specified, structural changes are written in the Astro hierarchical ECO format. IC Compiler can also read in hierarchical ECO files written by PrimeTime. Structural netlist changes are any what-if operations that modify the netlist structure, such as `insert_buffer` or `size_cell`. Once written, they can be applied to a cell in Astro with the following command, followed by ECO place and route operations:

```
auHECOByFile "lib" "cell" "heco_file.txt"
```

Note:

In IC Compiler, the hierarchical ECO flow became obsolete and has been replaced with a Tcl flow. In PrimeTime, you can use the `write_changes` command to generate an ECO output that is compatible for IC Compiler. For more information, see the *PrimeTime SI User Guide*.

If the `scheme` format is specified, the coupling separation in the current PrimeTime SI analysis is written as scheme commands for Astro. These coupling separations are applied in a PrimeTime SI analysis using the `set_coupling_separation` and `remove_coupling_separation` commands.

Once written out as a scheme file, you can load these separation directives to apply them to the currently active cell in Astro, followed by a “Detail Route Search & Repair” operation:

```
load "separations.scm"
```

Using Astro W-2004.12-SP1 or later for this flow is recommended. For more information about the `write_astro_changes` command, see the man page.

4

Context Characterization

Context characterization is the process of deriving the timing context of a subdesign from its environment in the parent design. The resulting information can be used for hierarchical timing analysis in PrimeTime or for synthesis or logic optimization in Design Compiler.

This chapter is described in the following sections:

- [Context Characterization Overview](#)
- [Deriving the Context of a Subdesign](#)

Context Characterization Overview

Context characterization captures the timing context of instances of subdesigns in the chip-level timing environment. The timing context of an instance includes clock information, input arrival times, output delay times, timing exceptions, design rules, propagated constants, wire load models, input drives, and capacitive loads.

Use context characterization for the following purposes:

- To set the timing constraints of a subdesign during synthesis or logic optimization in Design Compiler.
- To perform timing analysis hierarchically in PrimeTime while observing chip-level timing constraints.

Setting Synthesis or Optimization Constraints

The context characterization steps for setting synthesis or optimization constraints in Design Compiler are as follows:

1. Read the top design into PrimeTime.
2. Identify the subdesigns that require timing optimization.
3. Characterize the timing context for each subdesign.
4. Generate a Design Compiler script containing the context information.
5. Read the subdesign into Design Compiler.
6. In Design Compiler, read the script generated in step 4.
7. Perform module-level optimization of the subdesigns.

Performing Subdesign Timing Analysis

The context characterization steps for performing subdesign timing analysis in PrimeTime are as follows:

1. Read the design into PrimeTime.
2. Identify the subdesigns for timing analysis.
3. Characterize the timing context for each subdesign.
4. Generate a PrimeTime script containing the timing assertions.

5. Read a subdesign into PrimeTime.
6. Read the script generated in step 4.
7. Analyze the timing of the subdesign.

Deriving the Context of a Subdesign

The `characterize_context` command derives the timing and environment context from instances of subdesigns in the timing environment. The context of an instance is defined as

- Waveforms of the clocks affecting the instance
- Input arrival times
- Output required times
- Timing exceptions
- Design rules
- Logic constants
- Case analysis
- Wire load models
- Input drives
- Capacitive loads

Use the `characterize_context` command with the `write_physical_annotations` and `write_context` commands to export timing and physical information for a block from the chip-level environment.

The `characterize_context` command does not capture delays and parasitics annotated on the internal nets of the instance being characterized. To capture this information, use the `write_physical_annotations` command without the `-boundary_nets` option. If you omit the options, PrimeTime characterizes all the information. For more information, see the `characterize_context` man page.

To generate the timing-related context information for instance I1 and I2, enter

```
pt_shell> characterize_context -timing {I1 I2}
```

To generate the design rule and the logic constant information from the context of instance I2, enter

```
pt_shell> characterize_context -design_rules I2 -constant_inputs
```

Clock Information

PrimeTime records all clock signals that affect the instance being characterized. Clock information includes signals that feed an input pin of the instance and signals that have sources within the instance. PrimeTime derives information about all clocks driving registers or ports that launch data signals to or receive data signals from the instance.

PrimeTime characterizes the clock waveform, clock signal latency, and the clock uncertainty information for each clock being characterized. PrimeTime does not characterize propagated skew information for clocks feeding input pins of the instance being characterized.

The `write_context` command writes characterized clock information as a series of `create_clock`, `set_clock_latency`, and `set_clock_uncertainty` commands.

Input and Output Delay Times

PrimeTime characterizes arrival times of data signals at input pins of the instance being characterized and their launching clocks. PrimeTime derives required times of data signals at output pins and their related clocks.

The `write_context` command writes input arrival times as `set_input_delay` commands and writes output required times as `set_output_delay` commands.

Point-to-Point Timing Exceptions

PrimeTime derives point-to-point timing exceptions that affect the instance being characterized. The derived exceptions include

- False paths
- Multicycle paths
- `max_delay` and `min_delay` exceptions between clocked startpoints and endpoints

The `characterize_context` command calculates arrival and required times along with port capacitances so that the characterized design has the same timing as the chip-level design.

PrimeTime writes point-to-point exceptions as a series of `set_false_path`, `set_multicycle_path`, `set_max_delay`, and `set_min_delay` commands, depending on their type.

Constant Logic Values on Inputs

The `characterize_context` command derives constant logic values that are propagated to input pins of the instance being characterized.

PrimeTime derives logic constant values as logic constants and case analysis values as case analysis. For more information about case analysis, see the chapter on that topic in the *PrimeTime Fundamentals User Guide*.

Input Drive Strength and Port Capacitance

PrimeTime derives the following input drive strength and port capacitance information:

Driver information

The drivers of input pins of the instance being characterized are characterized. If the input pin is driven by a port that has a linear drive specified, this drive is also characterized. Drive information is written as `set_drive` and `set_driving_cell` commands.

Port capacitance information

Both the pin capacitance and the wire capacitance on input and output pins are characterized. Pin capacitance is written as a `set_load` command. Wire capacitance is characterized as a fanout number that derives the number of external loads at the port. This number is used together with the wire load model to estimate the boundary net capacitance. If the wire load mode is `enclosed` or `segmented`, each pin of the instance being characterized has a wire load model specified for it, depending on the enclosing hierarchy of the boundary net to which it is connected.

The fanout number is saved as a `set_port_fanout_number` command for PrimeTime and as a `set_load -fanout_number` command for Design Compiler. The wire load model is saved as a `set_wire_load_model` command for PrimeTime and Design Compiler. In addition to the fanout number, PrimeTime sets a fixed wire capacitance on the port in certain situations to ensure that the pin-to-pin wire capacitance of the boundary net after characterization is the same as that in the top-level design before characterization. This fixed value is written as a `set_load` command.

Wire Load Models

The `characterize_context` command derives the wire load mode and model for characterized instances. PrimeTime always inherits the top-level wire load mode from subdesigns. The wire load model for subdesigns is determined as follows:

- If the top-level mode is “top,” the subdesigns inherit the top-level wire load model.

- If the top-level mode is “enclosed” or “segmented,” the subdesigns inherit the wire load model from the enclosing hierarchy of the net or net segment.

Design Rule Checks

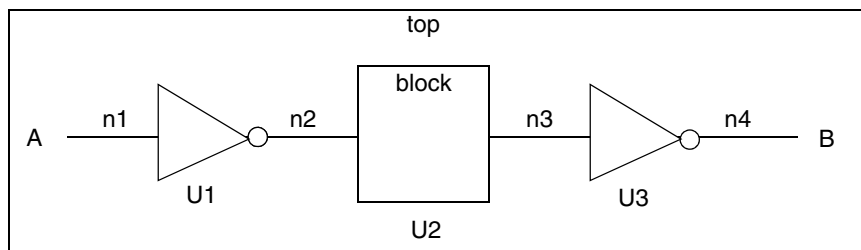
The `characterize_context` command derives design rule checks such as `max_capacitance`, `min_capacitance`, `max_fanout`, `min_fanout`, `max_transition`, and `min_transition` from their context. It also derives `fanout_load` for output pins of the instance being characterized. Each design rule check is saved by means of the corresponding command; for example, `max_fanout` is saved as a `set_max_fanout` command.

Annotated Delays and Parasitics

The `characterize_context` command does not capture delays and parasitics annotated on the internal nets of the instance being characterized. Use the `write_physical_annotations` command to capture this information.

Figure 4-1 shows a simple combinational subdesign called `block`. Script 1 manually sets the port interface attributes and Script 2 starts characterization.

Figure 4-1 Simple Combinational Design



Setup Script

```

current_design top
create_clock -name clk -period 10
set_input_delay 1.0 -clock clk A
set_output_delay 2.0 -clock clk B

```

Script 1

```

current_instance block
create_clock -name clk -period 10
set_driving_cell -lib_cell INV {C}
# derive driving cell information
set_input_delay 3.3 C -clock clk

```

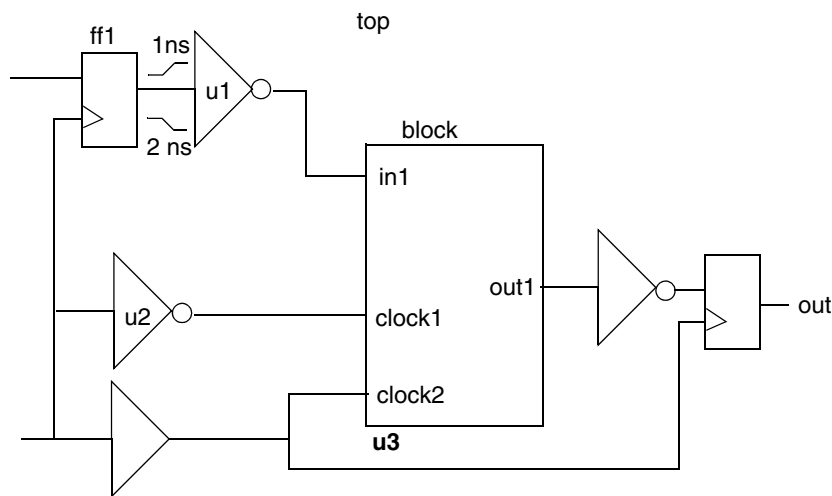
```
# arrival time of net n2 is 3.3
set_load 1.3 D
# load of U3 is 1.3
set_output_delay 3.5 D -clock clk
current_instance top
```

Script 2

```
current_design top
characterize_context U2
write_context -out U2.ptsh U2
```

Figure 4-2 shows a subdesign block in the sequential design called top. Script 1 sets port interface attributes manually and Script 2 invokes characterization.

Figure 4-2 Combinational Subdesign Block in Sequential Design



Script 1

```
current_design top
create_clock -period 10 -waveform {0 5} clock

current_instance block
create_clock -name clock -period 10 -waveform {0 5} clock1
create_clock -name clock_bar -period 10 -waveform {5 10}
clock2
# delay from ff1/CP to u5/in1 is 1.8
set_input_delay -clock clock 1.8 in1
# delay of u4 plus ff2 setup time is 1.2
set_output_delay -clock clock 1.2 out1
set_driving_cell -lib_cell INV -input_transition_rise 1 in1
\
-input_transition_fall 2 in1
```

```

set_driving_cell -lib_cell CKINV clock1
set_driving_cell -lib_cell CKBUF clock2
# outside load on out1 net
set_load 0.85 out1
current_instance top

```

Script 2

```

current_instance top
create_clock -period 10 -waveform {0 5} clock
characterize_context u5
write_context -out u5.ptsh u5

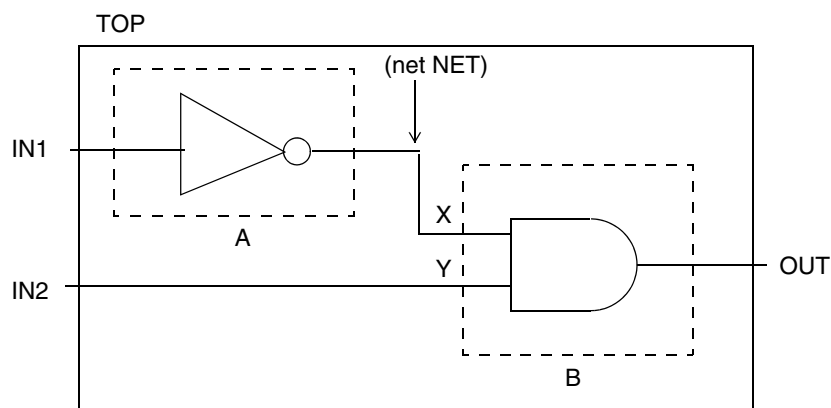
```

Input Delay and Port Capacitance

The `characterize_context` command calculates arrival and required times along with port capacitance so that the characterized design has the same timing as the chip-level design.

The design in [Figure 4-3](#) is the basis for the load and arrival time calculations using the `characterize_context` command shown in the following sections.

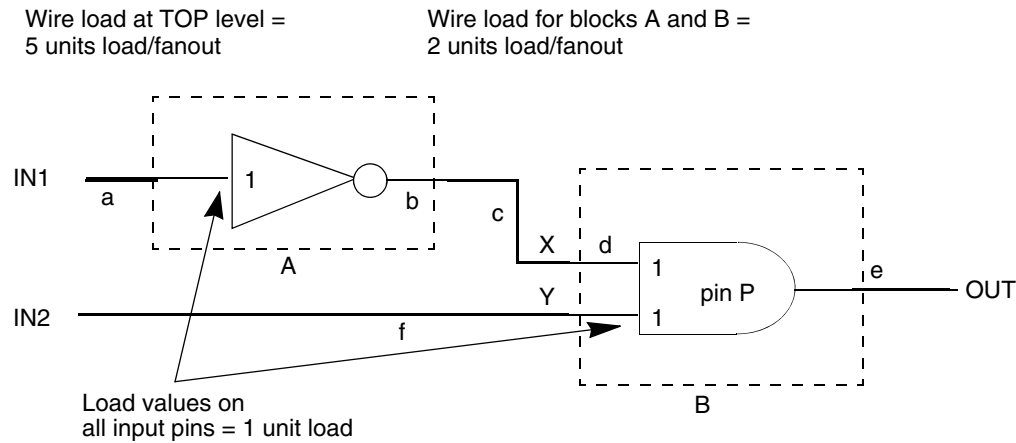
Figure 4-3 Example Hierarchical Design



Calculating Loads

Figure 4-4 shows how PrimeTime annotates the loads of ports and nets in the example in Figure 4-3 and provides some values for wire loads and input pin capacitance.

Figure 4-4 Design With Annotated Ports and Nets



In Figure 4-4, a through f represent wire segments used in the calculations. The example assumes a segmented wire load model that takes the interconnection net loads on the blocks into account and uses a linear function for the wire loads.

For the outside load for pin P in hierarchical block B, the calculation is

```
outside load =
  sum of the loads of all pins on the net loading P
  which are not in B
  + sum of the loads of all segments of net
  driving or loading P which are not in B
```

Each segment's load is

```
segment load =
  number of fanouts * wire load
```

In the following example, the outside load on input IN1 to block A is calculated by using 0 driving pins, a fanout count of 1 for segment a, and the TOP wire load of 5 loads per fanout:

```
load pins on driving net + load of segment a
= 0 + (1 * 5)
= 5
```

For the outside load on the output of block A, the calculation is

```
load pins on net
+ load of segment c
```

$$\begin{aligned}
 &+ \text{load of segment d} \\
 &= 1 \text{ (for load pin P)} \\
 &+ (1 * 5) \\
 &+ (1 * 2) \\
 &= 1 + 5 + 2 \\
 &= 8
 \end{aligned}$$

For each segment in the calculation, the local wire load model is used to calculate the load. The calculation for block A's output pin uses the TOP wire load of 5 loads per fanout for segment c and block B's wire load of 2 loads per fanout for segment d.

For the outside load on the output of block B, the calculation is

$$\begin{aligned}
 &\text{load pins on net} + \text{load of segment e} \\
 &= 0 + (1 * 5) \\
 &= 5
 \end{aligned}$$

For the outside load on the input pin X of block B, the calculation is

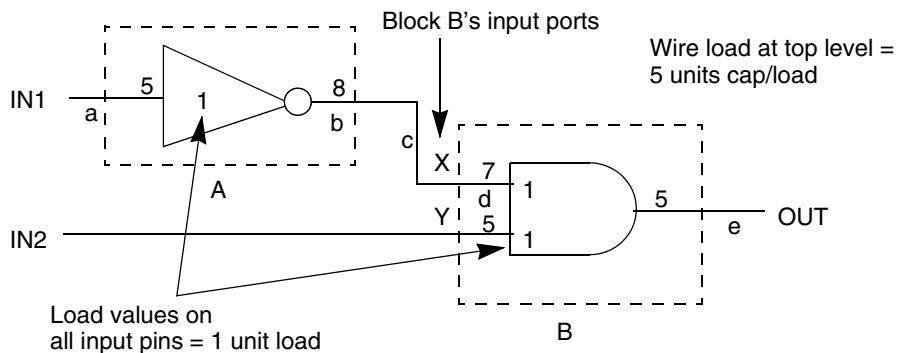
$$\begin{aligned}
 &\text{load pins on net} \\
 &+ \text{load of segment b} \\
 &+ \text{load of segment c} \\
 &= 0 + (1 * 2) + (1 * 5) \\
 &= 7
 \end{aligned}$$

For the outside load on the input pin Y, the calculation is

$$\begin{aligned}
 &\text{pin loads on driving net} + \text{load of segment f} \\
 &= 0 + 1 * 5 \\
 &= 5
 \end{aligned}$$

Figure 4-5 shows the same example with the outside loads annotated after characterization.

Figure 4-5 Design With Outside Loads After Characterization

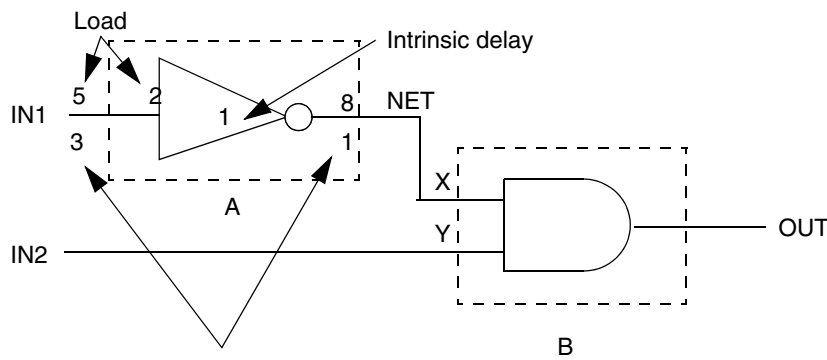


Calculating Input Delays

Because characterization provides accurate details of the outside loads, the input delays or path delays of input signals reflect only the delay through the last gate driving the port. They do not include the connect or transition delays. For example, the characterized arrival times on the input pins of block B are calculated from the delay to the pin that drives the port being characterized, without the transition or connect delays. This section describes the timing calculations for characterizing block B.

Figure 4-6 shows the example design annotated with default drive strengths and intrinsic delays of block A and signal IN1.

Figure 4-6 Design Annotated for Timing Calculations



For input pin X, the delay calculation is

```
drive strength at IN1 * (wire load + pin load)
+ intrinsic delay of A's cell
= 3 * (5 + 2 + 1)
+ 1
= 25
```

Writing Physical Information

The `write_physical_annotations` command writes physical information for a hierarchical block in a design. The exported information includes annotated net and cell delays using SDF. PrimeTime can also export annotated parasitics as a series of `pt_shell` and `dc_shell` (dctcl mode) commands.

To write the annotated delays on nets for the hierarchical cell U1, enter

```
pt_shell> write_physical_annotations -sdf U1.sdf \
-nets_only U1
```

To use the `-append` option to augment the script that PrimeTime generates by the `write_context` command with commands to import the generated delay and parasitics files, enter

```
pt_shell> characterize_context U2

pt_shell> write_context -format dtcl -out U2.dtc1 U2

pt_shell> write_physical_annotations -sdf U2.sdf \
        -parasitics U2.rc -format dcsh -append U2.dcsch U2
```

For more information, see the `write_physical_annotations` man page.

Reporting the Timing Context

The `report_context` command reports the timing context derived by the `characterize_context` command. If you do not specify an option, all context information is reported.

To report the timing-related context information for instance I1 and I2, use this command. The report shows clocks and their waveforms, point-to-point timing exceptions, input external delay, and output external delays.

```
pt_shell> report_context -timing {I1 I2}
```

To report the environment and design rule context information for I2, use this command. The report shows design rule checks, wire load models, driving cell information about input pins, and capacitive load on input and output pins of I2.

```
pt_shell> report_context -env -design_rules I2
```

For more information, see the `report_context` man page.

Generating Scripts for Characterized Contexts

The `write_context` command generates the timing of characterized contexts as a Design Compiler script or a PrimeTime script. Use this command to export context information to other tools. Use `write_context` with the `write_physical_annotations` and `characterize_context` commands to export timing and physical information for a block from the chip-level environment. If you do not specify any options, PrimeTime writes all context information.

To write the timing-related context information for instance I1 and I2 as a `dc_shell` script to standard output, enter

```
pt_shell> write_context -timing -format dcsh {I1 I2}
```

To write the environment and constant input context information for I2 as a PrimeTime shell script into a file called `des1.ptsh` file, use the following command:

```
pt_shell> write_context -environment \
           -constant_inputs -output des1.ptsh I2
```

For more information, see the `write_context` man page.

Removing Context Information

If you no longer need to report or write the context of an instance, you can delete the context information using the `remove_context` command. For the specified list of instances, the `remove_context` command deletes the timing context derived by the `characterize_context` command. If you do not specify an option, PrimeTime deletes all context information.

To delete the timing-related context information for instances I1 and I2, enter the following command:

```
pt_shell> remove_context -timing {I1 I2}
```

To delete the environment and design rule context information for instance I2, enter the following command:

```
pt_shell> remove_context -environment -design_rules I2
```

For more information, see the `remove_context` man page.

Limitations of Context Characterization

The following limitations apply to the `characterize_context` command:

Timing exceptions

To accurately constrain certain cases of timing exceptions involving registers, the `characterize_context` command can create virtual clocks and set input and output delays relative to that virtual clock instead of to the source register's clock. You might see these additional virtual clocks in the resulting constraint files.

The `characterize_context` command ignores

- Combinational timing exceptions starting outside the cell being characterized
- Timing exceptions specified with the `-through` option

The `characterize_context` command does not derive all combinational timing exceptions affecting paths in the instance being characterized. Combinational timing exceptions are exceptions defined between unclocked points in the design using the `set_max_delay` and `set_min_delay` commands.

Attributes in a design

The `characterize_context` command ignores clock latency or uncertainty and `max_time_borrow` attributes placed on a hierarchical boundary (generally not an issue because these attributes are usually placed on clocks and cells). The `characterize_context` command does not preserve `path_group` information when deriving output constraints for subdesigns; PrimeTime supports the default clock-based path groups.

Generated clock information

Generated clocks are expanded from the master clock and are characterized as separate clocks. They are saved by means of the `create_clock` command.

Modes

The `characterize_context` command does not characterize the mode information in the top design.

Back-annotation information

The `characterize_context` command does not derive back-annotation information, such as back-annotated delays and wire capacitance set on nets contained in the instance being characterized. Use the `write_physical_annotations` command in PrimeTime to do this.

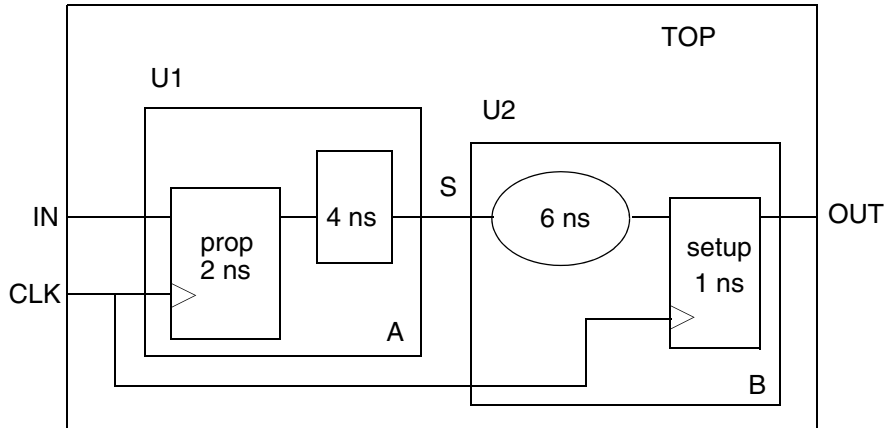
Timing budgets

The `characterize_context` command generates subdesign constraints on the basis of absolute path delays in the current design. Timing budgets are not usually needed when the outputs of modules are registered.

The `characterize_context` command generates constraints consistent with optimal timing budgets, if the design is already fully optimized and all paths have zero slack. The best design methodology in many cases is to use the `characterize_context` command to derive noncritical constraints and use timing budgets to constrain critical paths. Timing budgets give you more direct, repeatable control of optimization than characterized constraints. Timing budgets can also reduce the number of `characterize_context` and `compile` iterations necessary to converge on a coherent design hierarchy.

Figure 4-7 shows a design targeted to run at 50 MHz. The path delay through the circuit is 13 ns (including propagation delay through the first flip-flop and the setup time requirement of the second flip-flop). This design meets the timing requirement, but it is faster than necessary.

Figure 4-7 50-MHz Design



You can improve the circuit area by recompiling this design.

- In PrimeTime, use these commands:

```
pt_shell> create_clock CLK -period 20 -waveform {0 10}
pt_shell> characterize_context {A B}
pt_shell> write_context -out A.dcsh -format dcsh A
pt_shell> write_context -out B.dcsh -format dcsh B
```

- In Design Compiler, use these commands:

```
dc_shell> current_design A
dc_shell> include A.dcsh
dc_shell> compile
dc_shell> current_design B
dc_shell> include B.dcsh
dc_shell> compile
dc_shell> current_design TOP
dc_shell> report_timing
```

PrimeTime generates these constraints for design A:

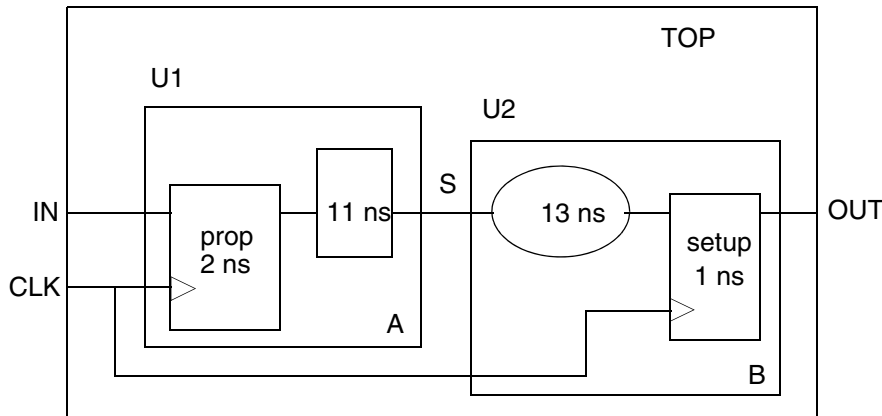
```
create_clock CLK -period 20 -waveform {0 10}
set_output_delay 7 -clock CLK S
```

PrimeTime generates these constraints for design B:

```
create_clock CLK -period 20 -waveform {0 10}
set_input_delay 6 -clock CLK S
```

When design A is compiled, Design Compiler can relax the path delay to 13 ns without violating the generated constraint. Similarly, when design B is compiled, Design Compiler can relax the path delay to 14 ns without violating the generated constraint. The resulting design, shown in [Figure 4-8](#), does not run at the required speed.

Figure 4-8 Design Resulting From Generated Constraint



If you repeat the `characterize_context` and `compile` commands on the design, `characterize_context` generates different constraints.

PrimeTime generates these constraints for design A:

```
create_clock CLK -period 20 -waveform {0 10}
set_output_delay 14 -clock CLK S
```

PrimeTime generates these constraints for design B:

```
create_clock CLK -period 20 -waveform {0 10}
set_input_delay 13 -clock CLK S
```

This resulting design might have the same constraints as the original design. It runs at speed, but it is unnecessarily large. When two blocks are characterized at the same time, then compiled in parallel, each block is overconstrained by the entire amount of negative slack on critical paths between the block and underconstrained by the entire amount of positive slack on noncritical paths between the blocks. To solve this problem, you can use the `characterize_context` command to compile the blocks repeatedly.

- In PrimeTime, use these commands:

```
pt_shell> create_clock CLK -period 20 -waveform {0 10}
pt_shell> characterize_context A -format dcsh -out A.dcs
```

Use the following commands for design B:

```
current_design TOP
read_ddc A.ddc
```

```
link TOP
characterize_context B -format dcsh -out B.dcsh
```

- In Design Compiler, use these commands:

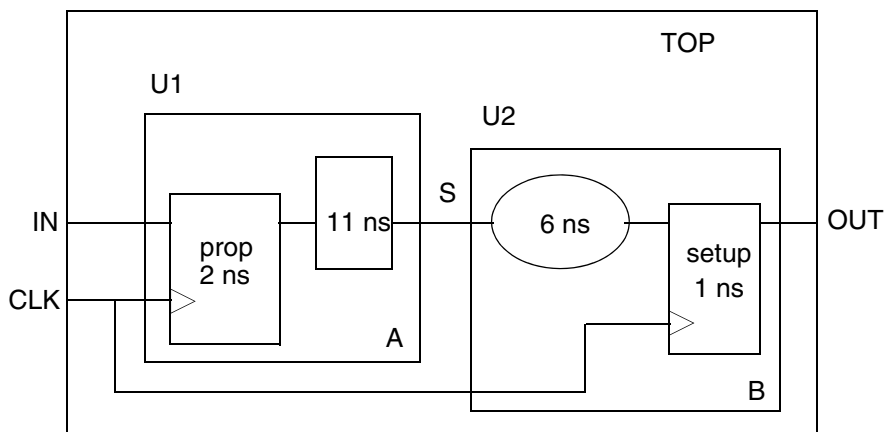
```
dc_shell> current_design A
dc_shell> include A.dcsh
dc_shell> compile
```

Use the following commands for design B:

```
current_design B
include B.dcsh
compile
current_design TOP
report_timing
```

Although this approach produces a smaller design that runs at speed, it might not produce the smallest design that runs at speed. In the preceding command sequence, the available slack is consumed by design A during the first compile. [Figure 4-9](#) shows the resulting design.

Figure 4-9 Design Resulting From Compiling Blocks Repeatedly



To realize the optimal design, you must set time budgets without using the `characterize_context` command. Instead, in Design Compiler, use these commands:

```
current_design A
create_clock CLK -period 20 -waveform {0 10}
set_output_delay 9 -clock CLK S
compile
current_design B
create_clock CLK -period 20 -waveform {0 10}
set_input_delay 11 -clock CLK S
compile
current_design TOP
report_timing
```


5

Advanced Analysis Techniques

There are several advanced timing analysis techniques that you can use in PrimeTime. This chapter explains the analysis techniques, commands, and options.

This chapter contains the following sections:

- [Time Borrowing in Latch-Based Designs](#)
- [Path-Based Timing Analysis](#)
- [High Capacity Mode](#)
- [Fast Analysis Mode](#)
- [Parallel Arc Path Tracing](#)
- [Support for Retain Arcs](#)
- [True and False Path Detection](#)
- [Asynchronous Logic Analysis](#)
- [Three-State Bus Analysis](#)
- [Fast Multidrive Delay Analysis](#)
- [Parallel Driver Reduction](#)
- [Data-to-Data Checking](#)
- [Interdependent Setup and Hold Pessimism Reduction](#)

Time Borrowing in Latch-Based Designs

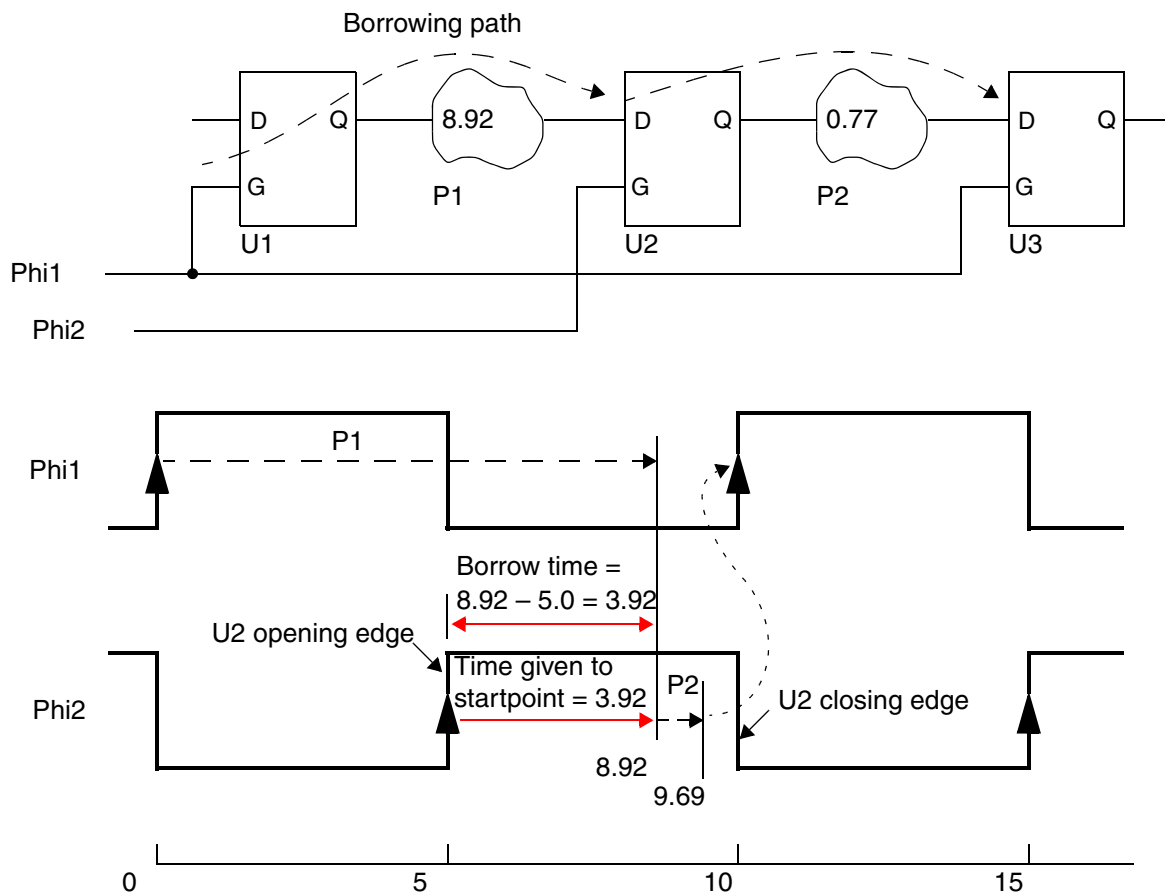
Transparent latches present unusual challenges for static timing analysis tools. A technique known as time borrowing (also known as “cycle stealing”) gives latch-based designs a distinct advantage over flip-flop based designs because a level-sensitive latch is transparent for the duration of an active clock pulse. This technique can relax the normal edge-to-edge timing requirements of synchronous designs. However, it is harder to control the timing of latch-based designs because of the multi-phase clocks used and the lack of “hard” clock edges at which events must occur.

Borrowing Time From Logic Stages

In a design using level-sensitive latches, a path can borrow time from the next logic stage by taking advantage of the fact that latches are transparent while the gate input is asserted.

Figure 5-1 shows latch-based stages using a simple two-phase clocking scheme.

Figure 5-1 Latch-Based Timing Paths



A design using level-sensitive latches allows a combinational logic path with a delay longer than the available cycle time as long as it is compensated by shorter path delays in subsequent latch-to-latch stages. For the two-phase design, the available time for latch-to-latch paths is half the clock cycle.

In [Figure 5-1 on page 5-2](#), U1, U2, and U3 are positive-level-sensitive latches (active when $G = 1$), and P1 and P2 are combinational logic paths. For now, assume a library setup time of zero for the latches and zero delay from D to Q in transparent mode. For positive-level-sensitive latches, PrimeTime uses the rising (opening) edge as the reference edge.

The figure shows path delays of $P1 = 8.92$ and $P2 = 0.77$. There might appear to be a violation at U2 because the data arrives at U2 after the rising edge of $\phi2$. However, because the U2 latch is transparent for 5 ns and P2 is less than that amount, path P1 can borrow the slack time (3.92 ns) from the path between U2 and U3. Therefore, the sum of P1 and P2 is 9.69, which is less than the required time of 10.00 at U3.

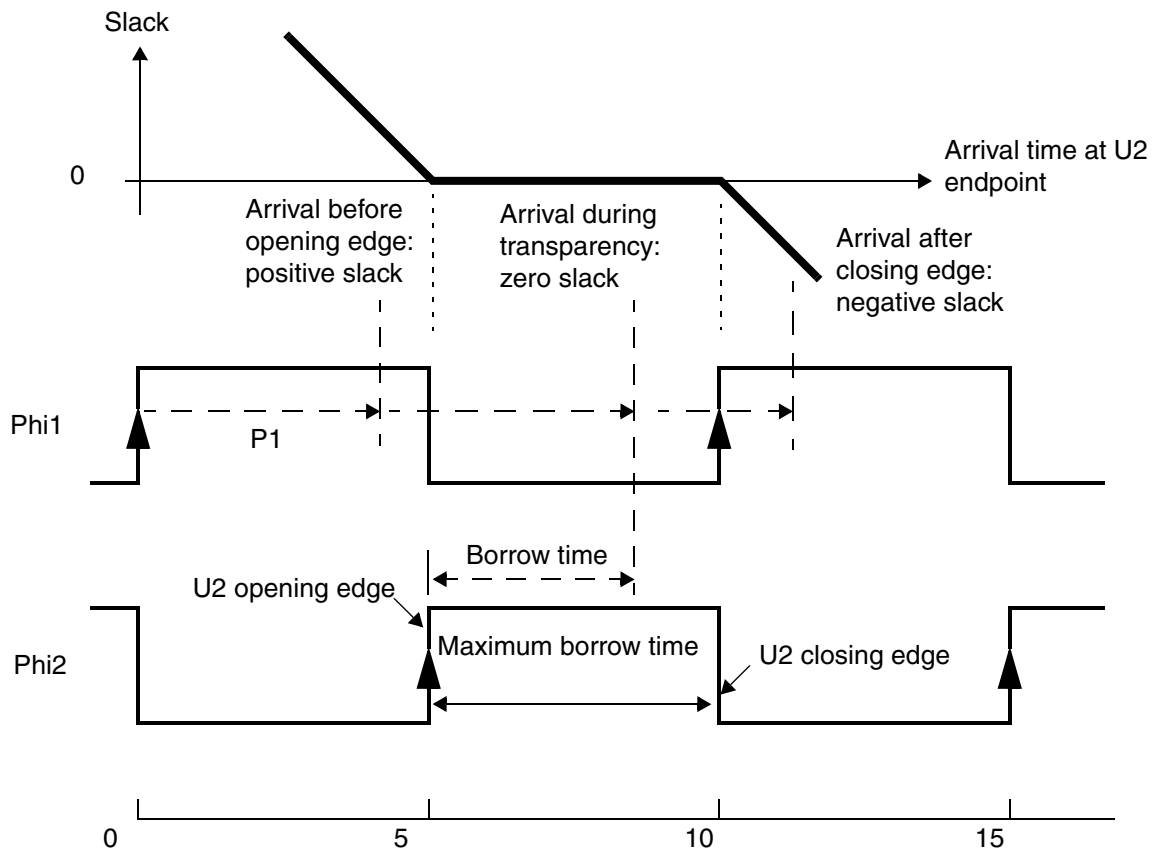
Latch Timing Reports

If the data signal arrives before the opening edge at the endpoint latch, PrimeTime models this behavior just as it would for a flip-flop. The opening edge of the clock (rising edge in this example) captures the data at the endpoint. The same clock edge launches the data at the startpoint of the next path.

On the other hand, if the data signal arrives while the latch is transparent (after the opening edge but before the closing edge at the endpoint latch), PrimeTime models the behavior at the next stage as a launch from the data pin of the latch, rather than from the clock pin. In this case, there is no timing violation and the slack is considered zero. The amount of time borrowed by the stage ending at the latch becomes the departure time for the next stage, subject to certain adjustments described in the next section, [“Maximum Borrow Time Adjustments” on page 5-7](#). A data signal arriving after the closing edge at the endpoint latch is a timing violation.

Figure 5-2 shows how PrimeTime would calculate and report slack for a range of combinational delays through P1, which would result in different arrival times at U1 (refer to the schematic in Example 5-1 on page 5-2). Figure 5-2 does not consider the effects of latch setup time, latency, uncertainty, and clock reconvergence pessimism removal.

Figure 5-2 Latch-Based Timing Path Slack Calculation



The following example shows how the `report_timing` command reports a timing path ending at a transparent latch with time borrowing. The “time borrowed from endpoint” and “time given to startpoint” statements in this report correlate with Example 5-1 on page 5-2.

```
*****
Report : timing
        -path short
        -delay max
        -max_paths 1
Design : time_borrow
*****
Wire Loading Model Mode: enclosed
```

Startpoint: U1 (positive level-sensitive latch clocked by Phi1)
 Endpoint: U2 (positive level-sensitive latch clocked by Phi2)
 Path Group: Phi2
 Path Type: max

Point	Incr	Path

-		
clock Phi1 (rise edge)	0.00	0.00
clock network delay (ideal)	0.00	0.00
U1/G (LATCH)	0.00	0.00 r
U1/Q (LATCH)	0.57	0.57 r
...		
U2/D (LATCH)	8.35	8.92 r
data arrival time		8.92
clock Phi2 (rise edge)	5.00	5.00
clock network delay (ideal)	0.00	5.00
U2/G (LATCH)	0.00	5.00 r
time borrowed from endpoint	3.92	8.92
data required time		8.92

-		
data required time		8.92
data arrival time		-8.92

-		
slack (MET)		0.00

Time Borrowing Information

Phi2 nominal pulse width	5.00
library setup time	-0.46

max time borrow	4.54
actual time borrow	3.92

Startpoint: U2 (positive level-sensitive latch clocked by Phi2)
 Endpoint: U3 (positive level-sensitive latch clocked by Phi1)
 Path Group: Phi1
 Path Type: max

Point	Incr	Path

-		
clock Phi2 (rise edge)	5.00	5.00
clock network delay (ideal)	0.00	5.00
time given to startpoint	3.92	8.92

U2/D (LATCH)	0.00	8.92 r
U2/Q (LATCH)	0.53	9.45 r
...		
U3/D (LATCH)	0.24	9.69 f
data arrival time		9.69
clock Phil (rise edge)	10.00	10.00
clock network delay (ideal)	0.00	10.00
U3/G (LATCH)	0.00	10.00 r
time borrowed from endpoint	0.00	10.00
data required time		10.00

-		
data required time		10.00
data arrival time		-9.69

-		
slack (MET)		0.31

Time Borrowing Information

Phil nominal pulse width	5.00
library setup time	-0.49

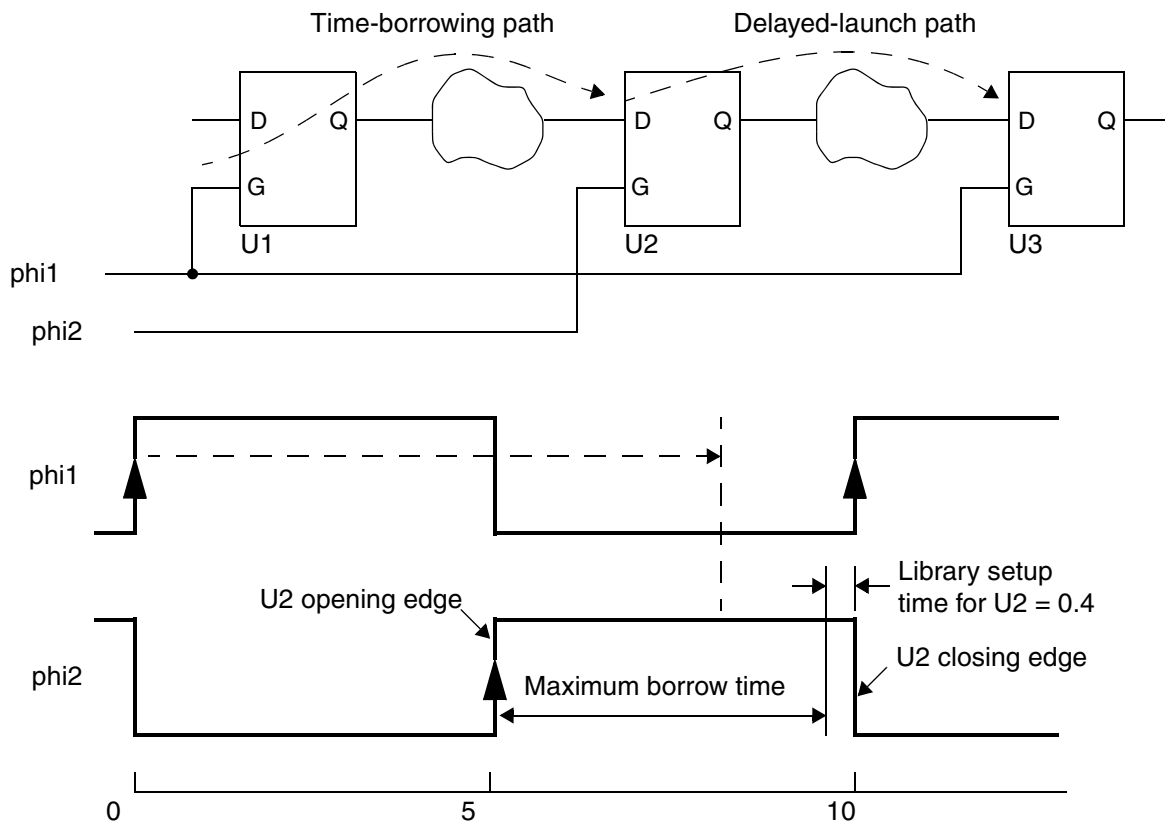
max time borrow	4.51
actual time borrow	0.00

For the U2 to U3 path, time must be added to compensate for the time borrowed, so PrimeTime adds 3.92 ns to the launch time of U2. This is reported in the second path's timing report. Because the P2 path has enough slack, neither path is in violation.

Maximum Borrow Time Adjustments

The maximum amount of time that can be borrowed at an endpoint latch is based on the clock pulse width (the time from the opening edge to the closing edge of the gate signal) as defined by the `create_clock` command, minus the library setup time of the latch, which is shown in [Figure 5-3](#).

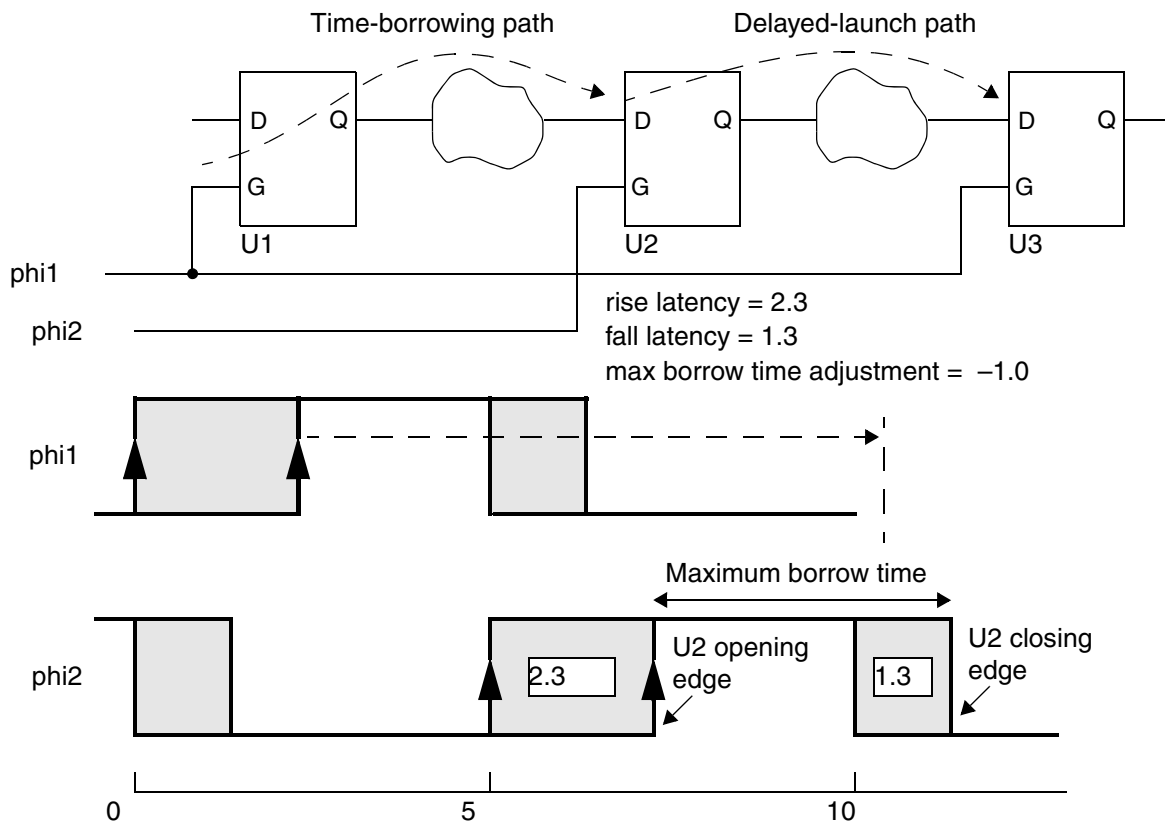
Figure 5-3 Maximum Borrow Time Reduced by Setup Requirement



For better accuracy, PrimeTime adjusts this amount further for the effects of clock latency, clock uncertainty, and clock reconvergence pessimism removal.

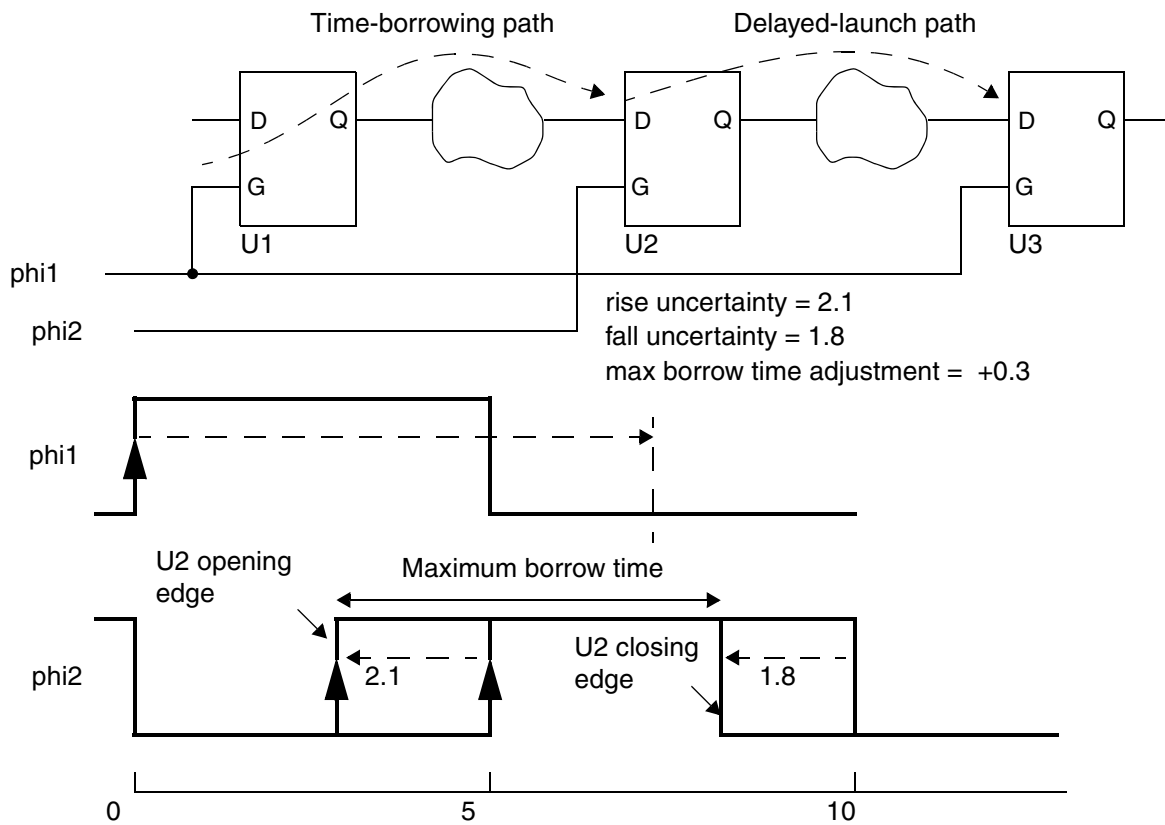
Clock latency can be defined with the `set_clock_latency` command or calculated by PrimeTime from propagated delays in the clock tree (enabled by `set_propagated_clock`). Latency values can be different for the rising and falling edges of the clock pulse, which can affect the pulse width and thus the maximum borrow time. See [Figure 5-4](#).

Figure 5-4 Maximum Borrow Time Adjustment for Latency



Clock uncertainty can be defined with the `set_clock_uncertainty` command. For a setup check, PrimeTime considers the earliest possible arrival of capture clock edges. Differences in uncertainty between rising and falling edges can affect the clock pulse width and maximum borrow time. See [Figure 5-5](#).

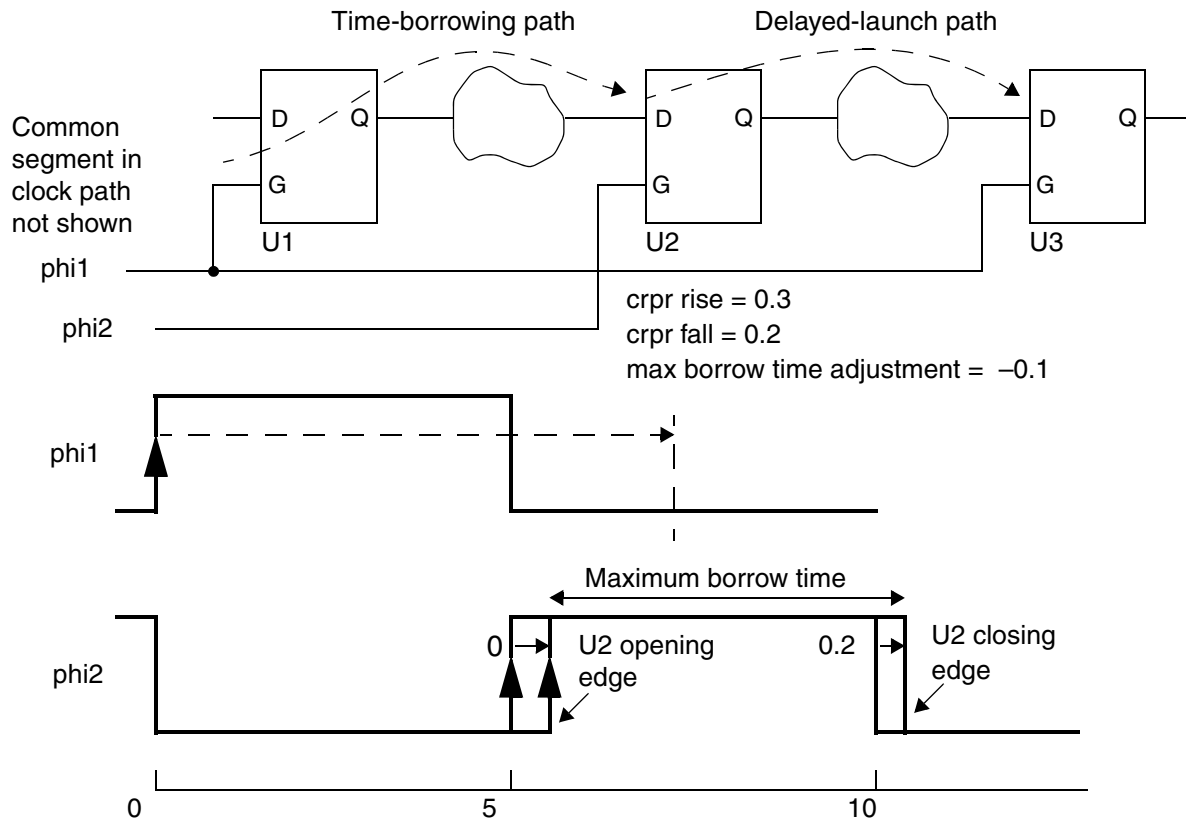
Figure 5-5 Maximum Borrow Time Adjustment for Uncertainty



Clock reconvergence pessimism removal (CRPR) is an analysis technique that corrects any inaccuracy resulting from a common segment in the launch and capture clock paths, as explained in the “Min-Max Analysis” chapter in the *PrimeTime Fundamentals User Guide*. You enable this feature by setting the `timing_remove_clock_reconvergence_pessimism` variable to `true`.

Application of CRPR shifts the clock edge times, like clock uncertainty. However, applying clock uncertainty makes the analysis more pessimistic, whereas applying CRPR makes the analysis less pessimistic, so the direction of the edge shift is in the positive direction rather than the negative direction. Differences in CRPR between rising and falling edges can affect the clock pulse width and maximum borrow time, which is shown in [Figure 5-6](#).

Figure 5-6 Maximum Borrow Time Adjustment for CRPR



To calculate the maximum allowable borrow time, PrimeTime starts with the clock pulse width and then adjusts it for the applicable effects of clock latency, clock uncertainty, clock reconvergence pessimism removal, and library setup time of the endpoint latch. The `report_timing` command reports the clock pulse width and the adjustments as follows:

Time Borrowing Information

```

-----
CLK nominal pulse width           5.00
clock latency difference          -1.00
clock uncertainty difference       0.30
CRPR difference                   -0.10
library setup time                -0.40
  
```

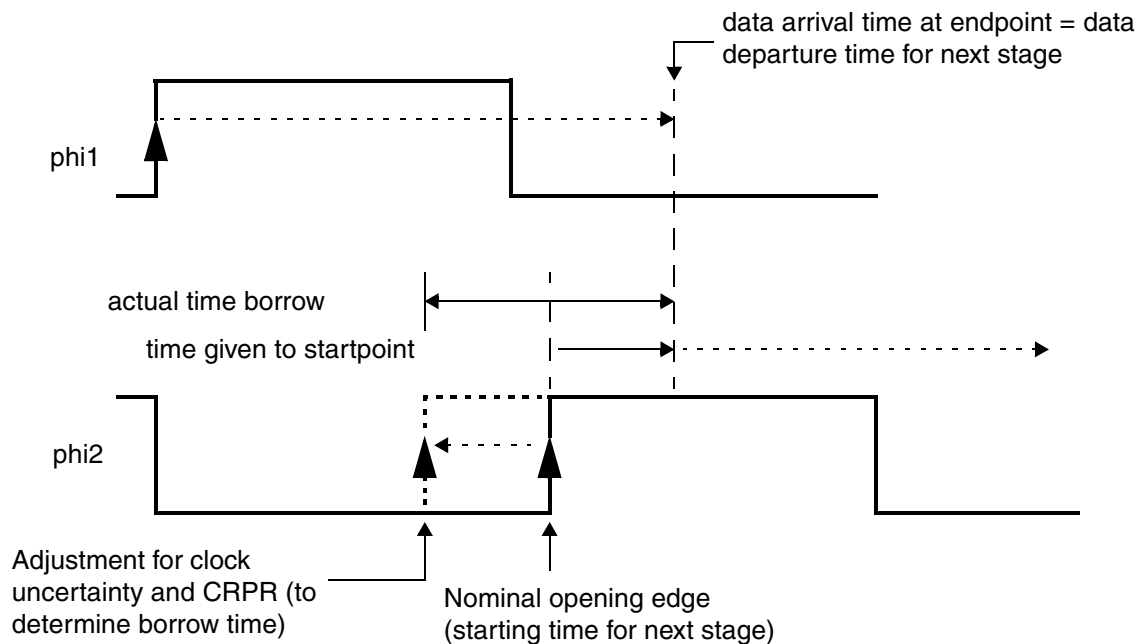
```
-----
max time borrow                               3.80
-----
...
```

Time Borrowed and Time Given

PrimeTime calculates the amount of time borrowed in relation to the arrival time of the opening clock edge at the latch. PrimeTime first adjusts the arrival time of the opening edge to account for path-specific effects of both uncertainty and CRPR (if applicable). Then it compares the adjusted value to the signal arrival time at the data pin to determine the amount of time borrowed at the path endpoint, if any.

If uncertainty and CRPR exist for the opening edge of the latch, the time borrowed is different from the amount of time given to the startpoint of the next stage. To determine the time given to the startpoint, PrimeTime subtracts the uncertainty and CRPR adjustments. This subtraction is necessary in transparent mode to make the launch at the next stage occur precisely when the signal arrives at the data pin, as illustrated in [Figure 5-7](#). When the `timing_early_launch_at_borrowing_latches` variable is disabled, the data arrival and launch times are not identical, owing to the deliberate application of a late clock latency to launch the next stage. This mode is recommended when CRPR is enabled. Note that the CRPR adjustment to the time given to the startpoint of the next stage is not applied in this mode. For more information, see the man page for this variable.

Figure 5-7 Borrow Time and Time Given to Startpoint in Transparent Mode



The `report_timing` command reports the amount of time borrowing and uncertainty/CRPR adjustments as follows:

Time Borrowing Information	

CLK nominal pulse width	5.00
clock latency difference	-1.00
clock uncertainty difference	0.30
CRPR difference	-0.10
library setup time	-0.40

max time borrow	3.80

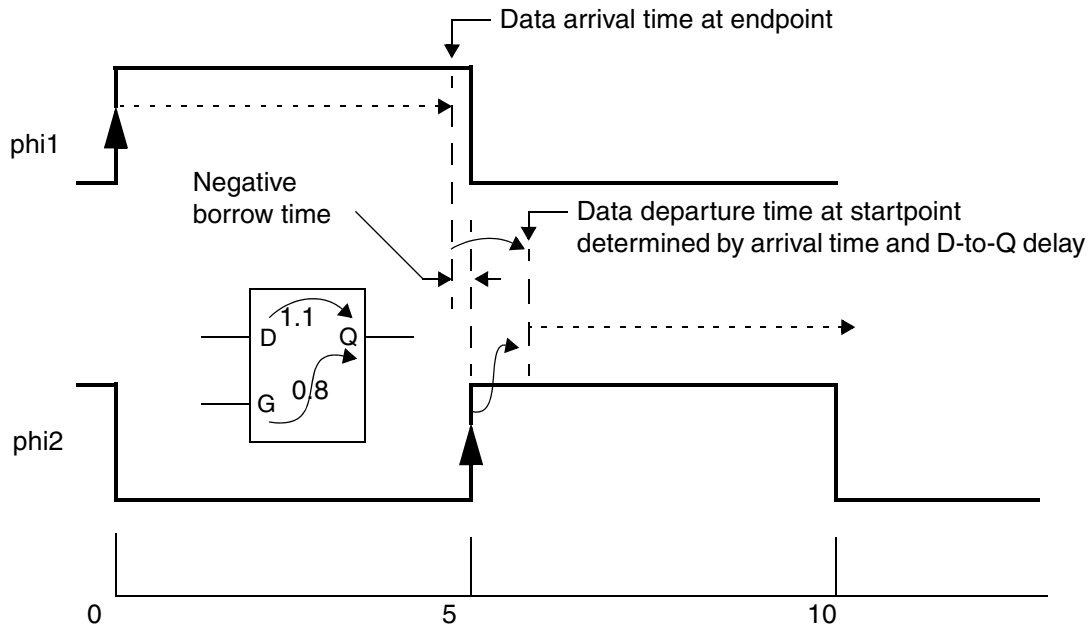
actual time borrow	3.40
open edge uncertainty	-2.10
open edge CRPR	0.30

time given to startpoint	1.60

In most cases, data arrival before the opening clock edge results in no borrowing, whereas data arrival after the opening clock edge results in borrowing. When borrowing occurs, the arrival time minus the clock edge time is reported as “time borrowed from endpoint” from the perspective of the path segment ending at the latch. This same amount of time is reported as “time given to startpoint” from the perspective of the path segment starting from the latch. However, in certain cases, data arrival just before the clock edge can result in borrowing. This happens when the D-to-Q delay of the latch is more than its clock-to-Q delay, and the

arrival time is very close to the clock edge, as shown in [Figure 5-8](#). Under these circumstances, the data departure time for the next path is determined by the data arrival time and the D-to-Q delay, rather than the clock-to-Q delay of the no-borrowing case.

Figure 5-8 Negative Borrow Time



PrimeTime accounts for this condition by allowing time borrowing. The arrival time minus the clock edge time is a negative number, so the amount of borrowing is negative. The borrowed amount is reported as “time given to endpoint” from the perspective of the path segment ending at the latch, and “time borrowed from startpoint” from the perspective of the path segment starting from the latch. This amount of time cannot exceed the difference between the D-to-Q delay and clock-to-Q delay of the latch.

Limiting Time Borrowing

The `set_max_time_borrow` command limits time borrowing to a specified amount for all latch endpoints within a specified scope of the design. Use this command to set a more restrictive constraint on borrowing. At the specified latch endpoints, PrimeTime limits borrowing to the specified amount or to the default value determined by the adjusted pulse width, whichever is smaller.

If the `default_max_time_borrow` constraint is zero, no time borrowing is allowed and PrimeTime analyzes the latch like a flip-flop.

If the `default_max_time_borrow` constraint is negative, the data signal must be stable before the open edge of the clock. Use this feature to ensure that the enable input arrives before the clock (on a gated-clock latch, for example). For more information, see the `set_max_time_borrow` man page.

To show the `max_time_borrow` attributes in the design, use the `report_exceptions` command. To remove the maximum time borrow limit set on specified objects using `set_max_time_borrow`, use the `remove_max_time_borrow` command.

These commands set maximum time borrowing of 2.5 on all latches clocked by `clk`:

```
pt_shell> create_clock -period 10 -waveform {0 5} clk
pt_shell> set_max_time_borrow 2.5 find(clock,clk)
pt_shell> report_timing -to latch1/D
```

The `user_max_time_borrow` constraint (from the `set_max_time_borrow` command) effectively reduces the pulse available for time borrowing from 5.0 to 2.5:

Time Borrowing Information	
user max_time_borrow	2.50
max time borrow	2.50
actual time borrow	0.00

The time borrowed from an endpoint is usually the same as the time given to the next startpoint when the endpoint and startpoint are both the D pin of the same latch. There can be a small difference due to uncertainty and CRPR adjustment of the opening edge of the latch. Sometimes, if the borrowing at D is small, the most critical path from the latch still comes from G; however, if you do the explicit report from D, then the time given is the same as the time borrowed.

Path-Based Timing Analysis

Static timing analysis tools are designed to be pessimistic in certain respects to ensure detection of all timing violations. For example, by default, PrimeTime considers both the worst arrival time and worst slew among all signals feeding into a path, even if the signal with the worst arrival time is different from the one with the worst slew.

Control the behavior of PrimeTime with respect to worst slew with the `timing_slew_propagation_mode` variable. When this variable is set to its default of `worst_slew` a possibly pessimistic but safe analysis is given. Setting this variable to `worst_arrival` gives a more accurate but possibly optimistic analysis.

Although the worst arrival mode reduces the pessimism for the critical path, non-critical paths may have optimism for highly slew-sensitive paths. For more information, see the man page for this variable or the description in the *PrimeTime Fundamentals User Guide*.

A better method to reduce the pessimism for critical paths is to analyze those paths in isolation from other paths. This method is called path-based timing analysis. When recalculating a timing path, PrimeTime propagates the edge along each path of interest, ignoring slews from side arcs along the path, and it performs delay calculation to compute the path-specific slack. Thus, path-based analysis recalculates the timing in a timing path without considering outside paths that might otherwise affect the arrival times and slew values used in the calculation. The command annotates the path collection with new timing information such as arrival times, slews, and slack.

Timing path effects, such as clock reconvergence pessimism removal and crosstalk effects (delta delays), are also recomputed specifically to that path. Path-based analysis is useful when there are only a few violations remaining in the analysis, and you want to find out whether these violations are caused by pessimistic analysis of arrival and slew times.

One important property of path-based analysis is that the slack ordering of paths can change due to the path recalculation. In other words, the most critical path before recalculation may not be the most critical path after recalculation. Because of this property, multiple paths to an endpoint may need to be recalculated to determine the true post-recalculation critical path. It cannot be emphasized enough that recalculating the single worst failing path to an endpoint does not provide the worst recalculated slack to that endpoint.

There are two ways to recalculate a specific path or set of paths. One way is by using the `-pba_mode path` option of the `report_timing` or `get_timing_paths` command. No path search is performed to determine whether those paths are truly the worst recalculated paths. Here is an example of the path-specific recalculation approach:

```
pt_shell> report_timing -pba_mode path
```

In this example, the single worst path in each path group is obtained from the design, then recalculated, then reported. No searching is performed for other paths that might have worse slack after recalculation. It is possible that there are subcritical paths that would have worse slack after recalculation than the reported paths. This method is useful for quickly gaining an idea of how much improvement is provided by path-based analysis without the runtime of a full path search. The slack results either match those from a full path search or are slightly optimistic.

The second way is to perform an exhaustive path search for the worst recalculated timing. You do this by using the `-pba_mode exhaustive` option of the `report_timing` command. PrimeTime performs an exhaustive recalculated path search, recalculating as many paths as necessary to ensure that the paths returned are the worst recalculated paths that meet the specified options.

To ensure optimal path search runtime, the following usage is recommended:

- Use the `-pba_mode exhaustive` option with the `-slack_lesser_than` option to keep the search bounded.
- Perform an exhaustive path search only when `-pba_mode path` shows that the slacks are reasonably close to the desired threshold.
- Ensure that the reporting is as specific as possible. If reports are desired only for specific path groups or startpoints and endpoints, specify the required reporting options.
- Enable worst parallel-arc reporting by setting the `timing_report_use_worst_parallel_cell_arc` variable to `true`. For more information about this variable, see the man page.

To see the critical path in each path group while applying exhaustive path-based recalculation, use the following syntax:

```
pt_shell> report_timing -pba_mode exhaustive -slack_lesser_than 0
```

In the example above, if no paths are reported with negative slack, the design has no timing violations.

Path-based recalculation can also be combined with other timing path selection options, such as `-from/-through/-to` or `-max_paths/-nworst`. For `-pba_mode path`, the `-slack_lesser_than` option applies to the original slack used to obtain the paths. The reported recalculated slack might be improved.

When using `-max_paths` or `-nworst` with `-pba_mode exhaustive`, the path limits represent the desired path counts at the completion of the search. Additional paths are searched during the path search process to ensure a complete result. For example, to report the worst path to each violating endpoint with recalculation taken into account, use the following syntax:

```
pt_shell> report_timing -pba_mode exhaustive -nworst 1 \  
                -slack_lesser 0 -max_paths 1000
```

Multiple paths to each failing endpoint are searched to find the worst recalculated failing path.

The `report_timing -pba_mode exhaustive` option does not support other options, such as `-true`, `-justify`, `-start_end_pair`, and some other postprocessing options, such as the `-slack_greater_than` option.

You can also use the `get_timing_paths -pba_mode path` command to recalculate a collection of timing paths. When passed a collection of timing paths with this option, the collection is recalculated and returned. For example,

```
pt_shell> set paths [get_timing_paths -max_paths 10]
pt_shell> set recalc_paths [get_timing_paths -pba_mode path $paths]
```

After a set of timing paths have been recalculated, use the `report_timing` command to report the recalculated timing values of the path collection. For example,

```
pt_shell> report_timing $recalc_paths
```

The recalculated values are accessible only by the `report_timing` and `get_attribute` commands, not by other commands, such as the `report_constraint` command. However, you can use the `report_attribute` command to report the newly calculated attribute values directly inside the path collection.

The `timing_path` objects have an `is_recalculated` Boolean attribute that indicates whether the timing path has been recalculated. The `report_timing` command indicates whether a timing path is recalculated or not by having the text, (recalculated), appear in the Path Type header field for the path. For example,

```
*****
Report : timing
        -path full
        -delay max
        -max_paths 1000
Design : led
Version: D-2010.06
Date   : Tue Jul  6 12:28:38 2010
*****
Startpoint: c (input port)
Endpoint:   z2 (output port)
Path Group: default
Path Type:  max (recalculated)
```

The path-based analysis feature allows control of whether or not clock paths and latch-based borrowing paths are recalculated. The `pba_recalculate_full_path` variable controls whether the clock path should be recalculated. When the variable is set to its default of `false`, clock paths and borrowing paths are never recalculated. Only the data portion of the path is recalculated. When the variable is set to `true`, the clock path and borrowing paths are always recalculated.

The `-path_full_clock_expanded` option of the `report_timing` command indicates only that the clock path should be included in the timing report. This option does not control whether the clock path is recalculated or not. If the `pba_recalculate_full_path` variable is set to `true` and the `-path_full_clock_expanded` option is not specified, the clock path is obtained and recalculated; however, the expanded clock path is not shown in the report.

The `-path full_clock_expanded` option of the `get_timing_paths` command indicates that the clock path should be obtained and stored along with the timing path collection objects. If the `pba_recalculate_full_path` variable is set to `true` and recalculated timing paths are obtained directly with the `get_timing_paths -pba_mode` option, the clock path is recalculated and used to compute the path timing; however, the full clock path is not stored in the timing path collection objects. If a `timing_path` object previously obtained without the `-path full_clock_expanded` option is being recalculated and clock path recalculation is enabled with the `pba_recalculate_full_path` variable set to `true`, an error message is issued indicating that the clock path is needed for correct and consistent results. For more information about this capability, see the `pba_recalculate_full_path` variable man page.

Note:

When the `pba_recalculate_full_path` variable is set to `true`, path-based analysis recalculates the borrowing at transparent latches only if the `-trace_latch_borrow` option has been specified. Otherwise, the existing borrowing at the original path's startpoint is used.

The `timing_report_recalculation_status` variable facilitates debugging and eases the runtime concern. It works for both the `get_timing_path -pba_mode exhaustive` and `report_timing -pba_mode exhaustive` commands. For more information about this variable, see the man page.

The amount of time needed for the integrated path search depends on a few factors:

- Amount of slack improvement provided by recalculation
- Number of paths involved in the search (`-slack_lesser` than, `-from/-through/-to`)
- Number of paths to be returned to the user (`-max_paths` and `-nworst`)

In particular, large `-nworst` values can significantly increase the runtime of the search. The `pba_exhaustive_endpoint_path_limit` variable helps to prevent excessive runtime by terminating the exhaustive path searching for an endpoint once a specific limit on the per-endpoint number of paths to recalculate has been reached.

When you perform a timing update on the original design (for example, with an `update_timing` command), any timing path collections created by the `get_timing_paths` command are automatically deleted.

Setting Recalculation Limits

You can control the behavior of the recalculation limits in path-based analysis. Path-specific recalculation with the `-pba_mode path` option has a limit of 2000000 paths per recalculation command. Exhaustive recalculation with the `-pba_mode exhaustive` option has a paths-per-endpoint limit controlled by the `pba_exhaustive_endpoint_path_limit` variable, which defaults to 25000.

When either limit is reached, the limiting behavior is controlled by the `pba_path_recalculation_limit_compatibility` variable. When set to its default value of `true`, no further paths are obtained or recalculated when the limit is reached. For path-specific recalculation, the remaining paths beyond the limit are discarded. For exhaustive recalculation, the limited endpoint is not included in the search; however, any paths that are already found for that endpoint remain in the search.

When the `pba_path_recalculation_limit_compatibility` variable is set to `false`, graph-based analysis paths are used to fill in the recalculation results past the limit. For path-specific recalculation, the remaining paths beyond the limit retain their original non-recalculated graph-based timing. For exhaustive recalculation, any additional paths obtained for the limited endpoint retains their original nonrecalculated graph-based timing. In both cases, recalculated paths can be readily identified in the resulting timing reports with the recalculation flag in the path headers.

Composite Current Source (CCS) Receiver Model for Path-Based Analysis

For path-based analysis, PrimeTime uses the actual receiver models in the path being analyzed. For the cells that are side loads, the worst-case receiver models are used. During path-based analysis, receiver models are derived using the effective capacitance (`Ceff`) on the output of the load cell. You save this capacitance for path-based analysis by setting the `rc_cache_min_max_rise_fall_ceff` variable to `true` (the default is `false`).

Note:

The `rc_cache_min_max_rise_fall_ceff` variable is automatically set to `true` if you have enabled crosstalk analysis or if you have invoked the GUI prior to the `update_timing` command.

High Capacity Mode

The current trend in most of the new designs is an increase in the amount and complexity of core logic. This leads to an increase in size and complexity of the design netlist, constraints, parasitics, and so on. To analyze large amounts of data efficiently, PrimeTime and PrimeTime SI have an optional high capacity mode targeted for very large designs with clock reconvergence pessimism removal (CRPR) enabled. The high capacity mode reduces the peak memory footprint and makes runtime trade-offs between performance and capacity, while producing the same results as normal analysis. High capacity mode is compatible with all analysis flows, including flat and hierarchical flows; however, it is most useful with CRPR enabled.

Enable high capacity mode using the `-enable_high_capacity` option of the `set_program_options` command. No other changes are required in your existing scripts or flows. You must issue the `set_program_options` command before loading designs or libraries. For example, the following command enables high capacity mode with the default settings.

```
pt_shell> set_program_options -enable_high_capacity
Information: enabling high capacity analysis mode...
(PTHC-001)
```

When high capacity mode is enabled in PrimeTime, the tool temporarily stores data to the local disk partition that is specified using the `pt_tmp_dir` variable, whose default is the `/tmp` directory. Upon exiting the session, the disk space consumed is automatically released. For effective use of this mode, it is recommended that the available capacity of the local disk on the host machine should be at least ~2X as large as the physical RAM memory.

Set the level of capacity effort using the `sh_high_capacity_effort` variable, which must be set before running `set_program_options`. This variable provides simple heuristics for trade-off between capacity and performance; it does not impact the analysis results. The values available are `low`, `medium` (the default), and `high`. For example, to set the caching effort to high, use

```
pt_shell> set sh_high_capacity_effort high
```

To disable the high capacity mode and return to the default normal analysis mode, use the `-disable_high_capacity` option of the `set_program_options` command. The `sh_high_capacity_enabled` read-only variable monitors whether high capacity mode is enabled.

For more information about the high capacity mode, see the `set_program_options` man page.

Fast Analysis Mode

The fast analysis mode in PrimeTime provides increased performance when accuracy is less critical. This mode provides a single setting that configures PrimeTime and PrimeTime SI for faster performance for the many early analysis runs while maintaining reasonable accuracy. When accuracy is critical, such as during late stage timing and ECO closure runs, you should ensure that this mode is disabled. By default, fast analysis mode is turned off. Enable fast analysis mode at the beginning of your script by using the `-enable_fast_analysis` option of the `set_program_options` command.

Note:

You should turn on fast analysis mode before loading designs and libraries because enabling fast analysis might cause changes when loading them.

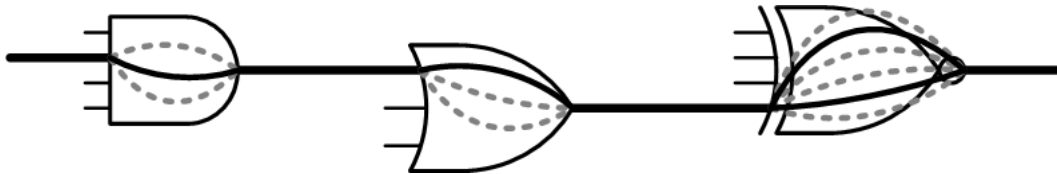
When you choose fast analysis mode, it gets higher priority over any other options or variables to provide higher performance. For example, it focuses default reporting on only violating paths. The `sh_fast_analysis_mode_enabled` read-only variable monitors whether fast analysis mode is enabled.

For more information, see the specific man pages.

Parallel Arc Path Tracing

To ensure that the `report_timing` command uses only the worst arc in each sense set of parallel arcs for path tracing, set the `timing_report_use_worst_parallel_cell_arc` variable to `true`. The variable is set to `false` by default. [Figure 5-9](#) shows the result if you enable this variable by setting it to `true`.

Figure 5-9 Worst Arc Used for Path Tracing



Only paths with the worst arc behaviors through the logic with a given rise and fall edge sequence are returned. The worst arc is the fastest arc for minimum delay tracing and it is the slowest arc for maximum delay tracing. Subcritical timing paths through other combinations of timing arcs with the same edge direction sequence are not returned. However, since timing arcs with different senses are still considered unique, different paths with unique rise and fall edge sequences through non-unate gates are still returned.

This might cut down substantially on the number of paths returned by large `-nworst` values, improving analysis efficiency. If the `-nworst` limit is not being reached, such as when reporting paths to a single endpoint, a lesser number of timing paths are returned. If `-nworst` has reached its limit, such as when reporting across an entire path group, the paths that are returned up to the limit might have a more varied topological exploration of the logic. This can help improve the efficiency of reporting scripts, bottleneck analysis, ECO scripts that are driven by `timing_path` collections.

This feature impacts both the `report_timing` and `get_timing_paths` commands when you specify an `-nworst` value greater than 1. Only parallel arcs of the same sense are affected. If there are different sense arcs in parallel, such as `positive_unate` and `negative_unate` across an XOR gate, they are still considered unique. This feature affects path tracing behavior at reporting time only. It does not affect the behavior of the `update_timing` command. You can change the value of the variable at any time without incurring a timing update penalty.

Support for Retain Arcs

PrimeTime can load retain arcs for timing models from library files, annotate the retain arcs from SDF input files, and report on these arcs. Retain arcs are similar to hold-check arcs and are typically used for modeling random access memory (RAM). They are defined between a clock pin and the data output of a RAM, and they are always defined in parallel with the parent arc, which is the ordinary or default delay arc between the same two pins. A retain arc does not generate an actual timing check during timing analysis, but is treated as another delay arc that is connected in parallel with its parent arc.

Clock-to-output retain arcs guarantee that the RAM output does not change for a specific interval of time after the clock edge. When the retain arc delay is less than its parent arc, the retain arc appears in a timing report for only the minimum delay paths. When the retain arc delay is longer than its parent arc, the retain arc can also appear in the maximum delay path report with no error messages or warnings.

To report all of the timing arcs for cells in a technology library, including retain arcs, use the `-timing_arcs` option with the `report_lib` command. Use the `check_timing -retain` command to check if the retain arc has a delay greater than its parent arc.

The `read_sdf` command supports retain arcs, but the default behavior of the `write_sdf` command does not support those arcs. To write out retain arcs, use SDF version 3.0 format, not the default version 2.1 format, by specifying the following syntax:

```
pt_shell> write_sdf -version 3.0 file
```

To map retaining information for arcs, use these functions:

- `min_rise_retain_delay`
- `min_fall_retain_delay`
- `max_rise_retain_delay`
- `max_fall_retain_delay`

True and False Path Detection

An advanced feature of PrimeTime is the ability to determine whether a specified path is a true path or false path. PrimeTime can also find the most critical true path in the design.

PrimeTime categorizes false paths into two main types: functional and delay-dependent.

Figure 5-10 shows an example of a functional false path that results from resource sharing. When mux1 is selected, the path through AND gate c_d is blocked.

Figure 5-10 Functional False Path Due to Resource Sharing

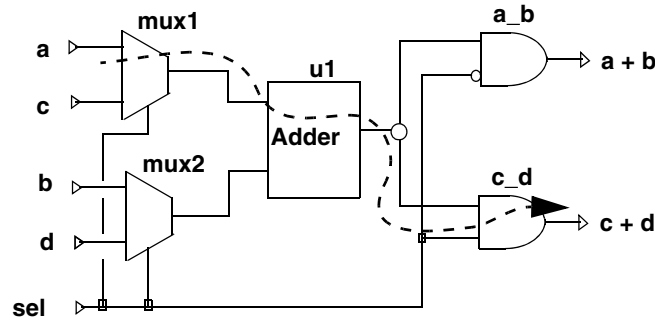
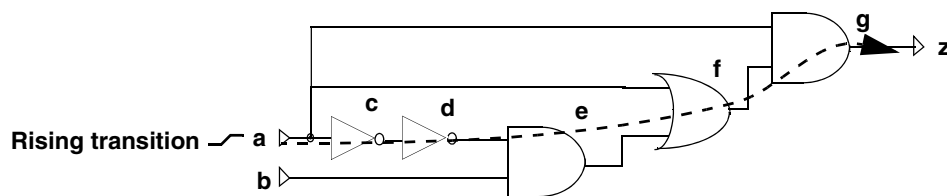


Figure 5-11 shows an example of a delay-dependent false path in carry-lookahead logic. The highlighted path (from a to g) is false for a rising transition at input a. The shorter direct path (from a, through f, to g) determines the longest true path.

Figure 5-11 Delay-Dependent False Path in Carry-Lookahead Logic



False path detection works in combination with case analysis. If you specify some pins in your design to be at a constant logic value, the false path detection features take these propagated logic constant values into account. For more information, see [“Justifying Paths” on page 5-24](#).

Reporting True or False Paths

When reporting a timing path, the `report_timing -justify` option uses a justification process to check whether the reported path is a true path or a false path. True path justification takes into account the logic function of the design and tries to find a set of input vectors that can sensitize the reported path.

If PrimeTime cannot find an input vector, it considers the path a false path. If it finds an input vector, it considers the path a true path and shows the input vector at the end of the report.

The `report_timing -false` option is similar to the `report_timing -justify` option except that it only reports false paths. Although this is a convenient way to identify the false paths in the design, do not use this command to create a set of `set_false_path` commands; if the delays change in the design, the false paths might become true paths.

Reporting True Paths

The true path reporting feature is an advanced algorithm that detects the most critical true path. Use the `-justify` or `-true` options to the `report_timing` command to incorporate functional information into the critical path computation. With `-true`, the `report_timing` command is less pessimistic.

`-justify`

Provides a quick means to determine whether any path shown is true.

`-true`

Runs a search algorithm to find the most critical true path.

A path is true if PrimeTime can find an input vector that sensitizes the path. Sensitization is determined using floating mode delay, in which all nodes are assumed to be in the X-state (unknown) before the application of an input vector.

The sensitization criteria for each gate on the path are as follows:

- The online signal is a controlling value, and there are no side inputs with controlling values that arrived earlier (all side inputs must be noncontrolling or later arriving).
- The online signal is a noncontrolling value. All other side inputs are noncontrolling, and no side input arrives later than the online signal.

For these criteria, PrimeTime defines arrival time as the time when a simulated value arrives to replace the X-state, not the structurally longest arrival time that PrimeTime typically computes.

Justifying Paths

If you suspect that the paths that PrimeTime is reporting are false, use the `report_timing -justify` command to generate a timing report. For most designs, the structurally longest path is a true path, so the `-justify` option finds an input vector that sensitizes the path. You can proceed with confidence knowing that PrimeTime is reporting the true delay (and that timing optimization is performed on the critical path).

The `-justify` option runs fast because it focuses on the single reported path (there is no searching and very little backtracking required). If PrimeTime finds an input vector, it displays it along with the path. If the path is false, PrimeTime reports it as such. You can see

more false paths by combining the `-justify` option with the `-nworst` option. If the `-justify` option reports that the structurally longest path is false, use the `-true` option to find the longest true path.

The `report_timing` command with the `-justify` option generates a report similar to this one.

```
pt_shell> report_timing -justify
```

```
*****
Report : timing
        -justify
Design : FP_SHR
*****
Operating Conditions:
Wire Loading Model Mode: top
```

A False path:

```
Startpoint: a (input port)
Endpoint: c_d (output port)
Path Group: default
Path Type: max
```

Point	Incr	Path

input external delay	10.00	10.00 r
a (in)	0.00	10.00 r
m1/Z (MUX21H)	1.00	11.00 r
u1/S (FA1)	1.00	12.00 r
c_d/Z (AN2)	1.00	13.00 r
c_d (out)	0.00	13.00 r
data arrival time		13.00
max_delay	15.00	15.00
output external delay	-10.00	5.00
data required time		5.00

data required time		5.00
data arrival time		-13.00

slack (VIOLATED)		-8.00

True-delay Input Vector

```
-----
( False Path )
-----
```

True-delay conclusions:

```
-----
1. No input-vector could be found to justify reported path.
```

The
search completed, so reported path is false.

Note:

In this example, the path reported starts with the rising transition at input a (see [Figure 5-11 on page 5-23](#) for more information).

Finding the Longest True Path

The `-true` option of the `report_timing` command runs a search algorithm to find the longest true path. For many designs, this algorithm runs very fast. However, some designs require a large amount of backtracking and the algorithm can take a long time to complete. PrimeTime provides a backtracking feature to control runtime.

In most cases, PrimeTime can find the longest true path fairly quickly using a low backtrack limit. Some designs require a high backtrack limit to find the real longest true path, but these designs are rare. The default backtrack limit is 1,000, which is a good compromise between runtime and accuracy.

The `report_timing` command with the `-true` option generates this report. In this example, particularly notice the lines that are in boldface.

```
pt_shell> report_timing -true
```

```
*****
Report : timing
        -true
Design : FP_SHR
*****
```

Operating Conditions:
Wire Loading Model Mode: top

A True path:

```
Startpoint: a (input port)
Endpoint: a_b (output port)
Path Group: default
Path Type: max
```

Point	Incr	Path
input external del	10.00	10.00
a (in)	0.00	10.00 r
m1/Z (MUX21H)	1.00	11.00 r
u1/S (FA1)	1.00	12.00 f
a_b/Z (AN2)	1.00	13.00 f
a_b (out)	0.00	13.00 f
data arrival time		13.00

max_delay	15.00	15.00
output external delay	0.00	15.00
data required time		15.00

data required time		15.00
data arrival time		-13.00

slack (MET)		2.00

True-delay Input Vector

```
-----
b (in)                r
cond (in)              f
a (in)                r
-----
```

True-delay conclusions:

```
-----
1. A path of length 13.00 (slack 2.00) was proven true.
The search completed, so there are no true paths that
have worse slack.
-----
```

Changing the Backtrack Limit

To change the backtrack limit, use the `true_delay_prove_true_backtrack_limit` variable.

When the search algorithm reaches this backtrack limit, it stops searching for true paths and saves the longest true path found. This path is proved true, but because the search reached the backtrack limit, longer true paths might exist but are not found. The proved true path provides a lower bound on the delay of the design. The algorithm then goes to a second phase in which it tries to prove paths false, starting with the structurally longest path and working downward. This second phase provides a tighter upper bound on the true delay of the design.

Generally, the prove-false phase reaches the backtrack limit. When both phases are complete, the true delay is between the lower (prove-true) value and the upper (prove-false) value.

Changing the Prove-False Backtrack Limit

To change the prove-false backtrack limit, use the `true_delay_prove_false_backtrack_limit` variable.

For almost all real designs, the search algorithm completes the search with the default settings and finds the longest true path. However, some designs are more difficult because they have millions of false paths close to the delay value of the longest true path. For these

designs, whether you are proving true (going up from 0.0) or proving false (going down from the structurally longest path), the algorithm eventually gets stuck because of all the false paths. At this point, the algorithm has useful information (a lower and upper bound on the true delay of the design). If you want the exact answer, set the backtrack limit very high so that the search algorithm continues processing until it completes the search.

When PrimeTime reaches the backtrack limit, the `report_timing` command prints conclusions at the end of the report that explain the results, as shown in the following example.

True-delay conclusions:

- ```

1. A path of length 42.00 (slack -42.00) was proven true.
 The search completed, so there are no true paths that
 have worse slack.

```

True-delay conclusions:

- ```
-----
1. A path of length 23.00 (slack -23.00) was proven true.
   However, The search reached the backtrack limit of 1000,
   so there may be other true paths that have worse slack
   but were not found. You may want to increase the prove-
   true limit by increasing
   true_delay_prove_true_backtrack_limit.

2. All paths of length 36.60 or longer were proven false.
   To get a tighter bound you can increase the prove-false
   limit by increasing
   true_delay_prove_false_backtrack_limit.
-----
```

Using the True Delay Function

The `-true_threshold` option of the `report_timing` command provides faster yes or no answers than the `-true` option. For example, if you want to know whether a path of length 18.0 or greater exists, define a true threshold of 18.0. Defining a true threshold accelerates the algorithm in two ways:

- It does not consider any paths shorter than the defined value.
- If it finds a path of this length or greater, it stops searching immediately.

The true delay algorithm works by trying to find an input vector that sensitizes the current path. The input vector applies rising or falling values at time zero to a design previously initialized to all X-state values. The time at which the last output transitions from X becomes the delay of the design for that vector. PrimeTime propagates the rising and falling values through the design using timed simulation and the floating delay sensitization criteria.

Long Path Example

In a carry-bypass adder, the long path of the carry chain is made into a false path through the addition of logic to provide a shorter path to the output (in the paths where the carry would propagate down the entire chain).

Examples

For a carry-bypass adder design, the `report_timing` command reports the carry chain as the longest path. In this example, particularly notice the lines that are boldface. To avoid getting this type of report, declare the long path to be a false path.

```
pt_shell> report_timing
```

```
*****
Report : timing
Design : false12cbp2
*****
```

```
Startpoint: s (input port)
Endpoint: x[11] (output port)
Path Group: default
Path Type: max
```

Point	Incr	Path

input external d	0.00	0.00 r
s (in)	0.00	0.00 r
U27/z1937 (*GEN*22	1.00	1.00 f
U257/z2167 (*GEN*21	1.00	2.00 f
U233/z2143 (*GEN*20	1.00	3.00 f
U234/z2144 (*GEN*205)	1.00	4.00 r
U239/z2149 (*GEN*185)	1.00	5.00 r
U108/z2018 (*GEN*48)	1.00	6.00 r
U199/z2109 (*GEN*270)	1.00	37.00 r
U200/z2110 (*GEN*46)	1.00	38.00 r
U167/z2077 (*GEN*129)	1.00	39.00 r
U1/z1911 (*GEN*283)	1.00	40.00 r
x[11] (out)	0.00	40.00 r
data arrival time		40.00
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00

data required time		0.00
data arrival time		-40.00

slack (VIOLATED)		-40.00

```
pt_shell> true_delay_prove_true_backtrack_limit = 1000000
1000000
```

```
pt_shell> report_timing -true
```

```
*****
Report : timing
      -true
Design : false12cbp2
*****
```

A True path:

```
Startpoint: s (input port)
Endpoint: y[11] (output port)
Path Group: default
Path Type: max
```

	Incr	Path

input external d	0.00	0.00 f
s (in)	0.00	0.00 f
U27/z1937 (*GEN*22	1.00	1.00 r
U248/z2158 (*GEN*6	1.00	2.00 f
U231/z2141 (*GEN*2	1.00	3.00 r
U232/z2142 (*GEN*21	1.00	4.00 f
U240/z2150 (*GEN*99)	1.00	5.00 r
U108/z2018 (*GEN*48)	1.00	6.00 f
U198/z2108 (*GEN*272)	1.00	20.00 f
U199/z2109 (*GEN*270)	1.00	21.00 r
U200/z2110 (*GEN*46)	1.00	22.00 f
U167/z2077 (*GEN*129)	1.00	23.00 r
U13/z1923 (*GEN*82)	1.00	24.00 f
y[11] (out)	0.00	24.00 f
data arrival time		24.00
max_delay	0.00	0.00
output external delay	0.00	0.00
data required time		0.00

data required time		0.00
data arrival time		-24.00

slack (VIOLATED)		-24.00
True-delay Input Vector		

c[11] (in)	r	
a[11] (in)	r	
d[11] (in)	f	
b[11] (in)	f	
c[10] (in)	r	
a[10] (in)	r	

```

d[10] (in)          f
b[10] (in)          f
d[9] (in)           r
b[9] (in)           r
c[9] (in)           f
a[9] (in)           f
d[8] (in)           r
b[8] (in)           r
c[6] (in)           f
a[6] (in)           f
c[5] (in)           r
a[5] (in)           r
c[4] (in)           r
a[4] (in)           r
d[4] (in)           f
b[4] (in)           f
c[3] (in)           r
a[3] (in)           r
d[3] (in)           f
b[3] (in)           f
c[2] (in)           r
a[2] (in)           r
d[2] (in)           f
c[0] (in)           r
a[0] (in)           r
d[0] (in)           r
b[0] (in)           r
c[1] (in)           f
a[1] (in)           f
d[1] (in)           r
b[1] (in)           f
s (in)              f
...
-----

```

True-delay conclusions:

- ```

1. A path of length 24.00 (slack -24.00) was proven true.
 The search completed, so there are no true paths that
 have worse slack.

```

---

## Asynchronous Logic Analysis

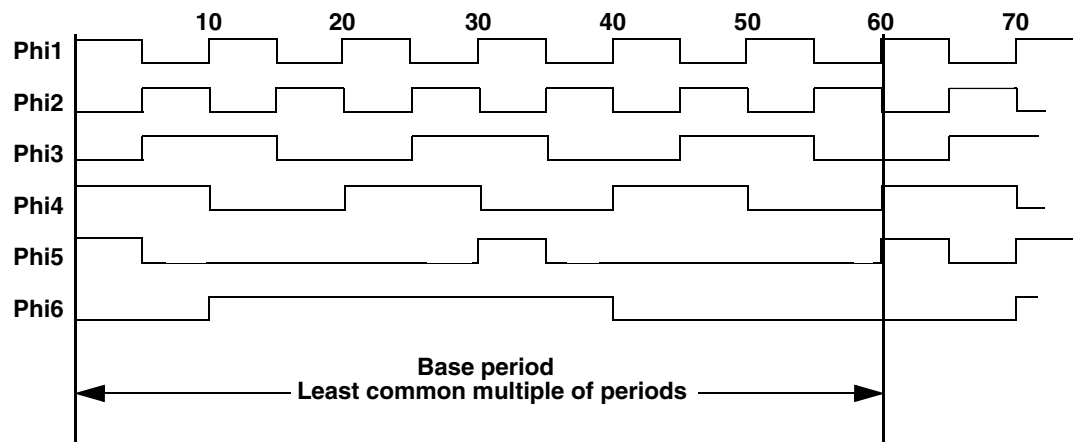
To simplify the timing verification designs with asynchronous logic, isolate the asynchronous logic into separate blocks, then disable the timing of these blocks.

PrimeTime does not support self-timed asynchronous logic where no global clock is used. Isolate this type of logic in a level of hierarchy and then use full-timing gate-level simulation to verify valid timing and functional capability.

PrimeTime can analyze designs:

- With no combinational feedback loops; loops containing flip-flops or latches are adequate (combinational feedback loops are automatically broken)
- Without unclocked memory elements, such as RS latches
- With a single clock or multiple clocks fanning in to each register clock pin
- With known and fixed phase relationship between the clocks at the start and end registers of every path (Interacting clocks must have a single base period over which all clock waveforms repeat—see [Figure 5-12.](#))

*Figure 5-12 Base Period of Clocks*



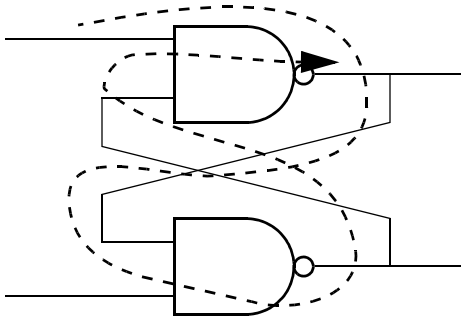


---

## Combinational Feedback Loop Breaking

A combinational feedback loop is a path that can be traced through combinational logic back to its starting point. [Figure 5-13](#) shows an example. To analyze such a path, PrimeTime must break the loop (stop tracing the path) at some point within the loop. Check a design for the presence of combinational feedback loops with the `check_timing -include loops` command.

*Figure 5-13 Combinational Feedback Loop*



By default, PrimeTime identifies each feedback loop and disables one of the timing arcs of the loop, such as the timing arc from one input to one output of a NAND gate in the loop. In some cases, this approach can result in some real paths not being reported because the paths are broken by the disabled arcs.

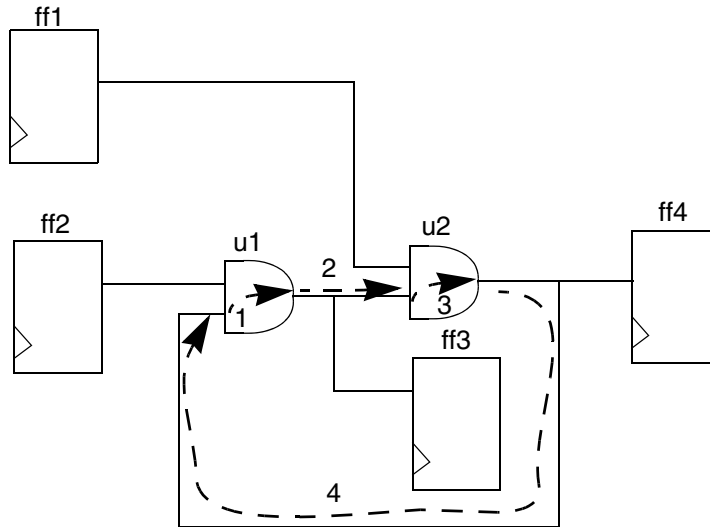
## Dynamic Loop Breaking

An optional technique called dynamic loop breaking guarantees that all valid paths of the design are reported. To enable dynamic loop breaking, set the `timing_dynamic_loop_breaking` variable to `true`. By default, the variable is set to `false` and PrimeTime does static loop breaking.

If the scope of the design that must be updated is large or if the loops are complex, using dynamic loop breaking might significantly increase the runtime or memory usage for a timing update.

Figure 5-14 shows an example where dynamic loop breaking helps to ensure that all paths get reported. In this example, no timing arc of the combinational loop can be broken without a valid path of the design also being broken. Arcs #1 and #4 cannot be broken because breaking them would break the valid path  $ff1 - u2 - u1 - ff3$ . Arcs #2 and #3 cannot be broken because breaking them would break the valid path  $ff2 - u1 - u2 - ff4$ .

Figure 5-14 Dynamic Loop Breaking



## Specifying Loop Break Points

You can have multiple places in a feedback loop that are broken. If you want a feedback loop to be broken at a different timing arc from the one selected by default, use the `set_disable_timing` command to explicitly break the loop at the desired point.

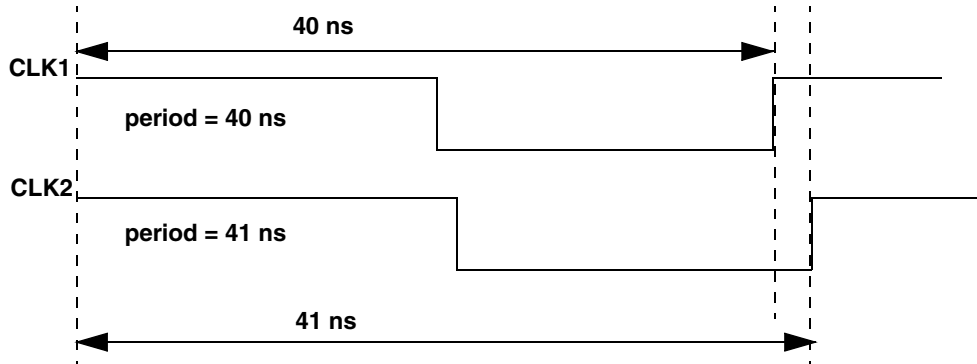
Design Compiler also uses loop breaking to analyze timing. The points at which Design Compiler and PrimeTime break a loop can be different, possibly leading to different timing results. If you want to ensure consistent loop breaking between the two tools, set the `timing_keep_loop_breaking_disabled_arcs` variable to `true` in PrimeTime. In that case, PrimeTime inherits the loop-breaking choices from the `.ddc` or `.db` file generated by Design Compiler. By default, this variable is set to `false`. After changing this variable setting, a timing update is necessary to see the change.

The report generated by `report_disable_timing` distinguishes between arcs disabled by PrimeTime and those disabled by inheritance from the `.ddc` or `.db` file. For more information, see the man page for the command.

## Unrelated Clocks

Sometimes a design has paths between unrelated clocks. Unrelated clocks have different frequencies that do not have a reasonable base period. PrimeTime attempts to find a base period and phase relationship anyway, which typically is not useful (see [Figure 5-15](#)).

Figure 5-15 Unrelated Clock Waveforms



The clocks shown in [Figure 5-15](#) do not expand to a reasonable base period, and the resulting setup requirement is quite restrictive. In this case, you might want to verify the timing with dynamic simulation rather than use PrimeTime.

To exclude asynchronous paths from static timing analysis to improve runtimes and avoid false violations, enter the following command:

```
pt_shell> set_false_path -from [get_clocks clk40] \
 -to [get_clocks clk41]
```

In some cases, you might know the required minimum and maximum path delays for the combinational logic between two registers of unrelated clocks. In these cases, specify the path delay constraint using the `set_max_delay` and `set_min_delay` commands for that path.

## Three-State Bus Analysis

By default, PrimeTime checks for setup and hold violations in three-state bus designs. It checks the worst delay path to ensure that the latched signals are stable and are not unknown (X) values. This ensures proper timing checks for transient bus contention and transient floating bus conditions.

PrimeTime considers that a disabling transition on a three-state cell can cause either a 1 or 0 delay on the output. By default, PrimeTime considers `three_state_disable` and `three_state_enable` arcs (as defined in the library) during path tracing. Although this is different from propagating an X value, the effect for static timing is the same as for propagating an X value.

---

## Limitations of the Checks

The three-state bus checks have these limitations:

- PrimeTime checks the potential for setup or hold errors due to bus contention or float conditions. PrimeTime does not check logical and power violations for bus contention or float conditions.
- The analysis is pessimistic; some reported violations might never happen because of state dependencies.
- In some simulators, Z does not propagate to X until after the charge decay time. PrimeTime does not model this effect; it uses the gate delay equations to propagate Z and X. This might be pessimistic compared to some simulators.

---

## Disabling the Checks

If you know that bus contention or floating buses do not occur in your design, disable these checks by setting these variables to `true`:

`timing_disable_bus_contention_check`

When you set this variable to `true`, propagation of maximum delay along `three_state_disable` timing arcs and minimum delay along `three_state_enable` arcs is disabled. These checks are valid only during transient bus contention. The default is `false`.

`timing_disable_floating_bus_check`

When you set this variable to `true`, propagation of minimum delay along `three_state_disable` timing arcs and maximum delay along `three_state_enable` arcs is disabled. These checks are valid only during floating bus conditions. The default is `false`.

---

## Bus Contention

Some designs rely on a bus configuration in which many three-state drivers control the bus. In most designs, no two drivers with different logical outputs can be simultaneously enabled at the steady state (when the enable pins of the three-state drivers assume their steady-state values for any clock cycle). When multiple drivers drive the same bus, it is called bus contention.

Although you can design a circuit so that no steady-state bus contention occurs, a design might contain transient bus contention conditions. Transient bus contention conditions occur during the transition of the bus control from one driver to another. During this short transient period, the logical value for the bus is unknown (X value) if the drivers contending for control of the bus are imposing conflicting logical values. For static timing analysis, setup checks ensure that this X value, assumed during transient bus contention, is not latched. The setup check is measured from the time the bus becomes stable.

---

## Floating Buses

A floating bus condition can arise for three-state bus configuration designs. A floating bus condition occurs when no driver is enabled. In this case, the bus immediately gets an unknown (X) value. For static timing analysis, hold checks ensure that this X value, assumed when the bus switches to the floating mode, is not latched. The hold checks are measured to the time the bus becomes floating.

---

## Three-State Buffers

Two timing arc types are used to describe timing behavior of three-state buffers. These timing arc types are defined in the library.

`three_state_disable` timing arc

Specifies the time the three-state pin takes to go from a high or low state to the high-impedance state.

`three_state_enable` timing arc

Specifies the time a three-state pin takes to go from the high-impedance state to a high state or low state.

## Performing Transient Bus Contention Checks

Figure 5-16 shows a circuit used to describe how PrimeTime performs transient bus contention checks and floating bus checks.

Figure 5-16 Circuit Example

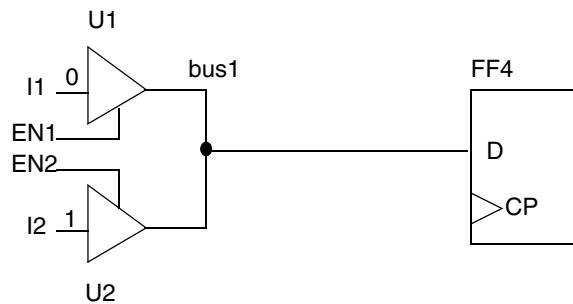
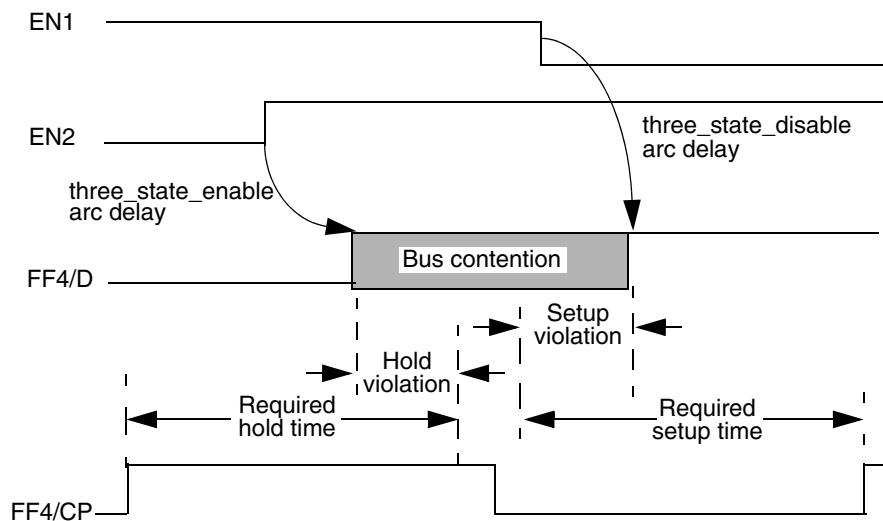


Figure 5-17 shows how PrimeTime performs transient bus contention checks.

Figure 5-17 Transient Bus Contention Check



In Figure 5-16, if EN1 turns off after EN2 turns on, there is a potential bus contention for some time. The signal arriving at FF4/D is X state until the contention is over and the new bus value propagates along the path. If I1=0 and I2=1, the waveforms for the circuit in Figure are as shown in Figure 5-17.

The setup violation for FF4/CP is seen only if the three\_state\_disable arc delay is considered. A transition to the Z state must be propagated as a possible transition to logic 0 or logic 1 to find all possible cases. A similar situation occurs for the hold check. The hold violation is seen only if the three\_state\_enable arc delay is considered.

This bus contention region is bounded by the minimum `three_state_enable` arc delay of any bus driver from one side and by the maximum `three_state_disable` arc delay from the other side. If you know that such bus contention regions can never occur, disable checking for both setup and hold violations that occur due to this bus contention region.

Disable the checks by setting the `timing_disable_bus_contention_check` variable to `true`, causing PrimeTime to ignore the `three_state_enable` arc delay for hold violation checking and the `three_state_disable` arc delay for setup violation checking.

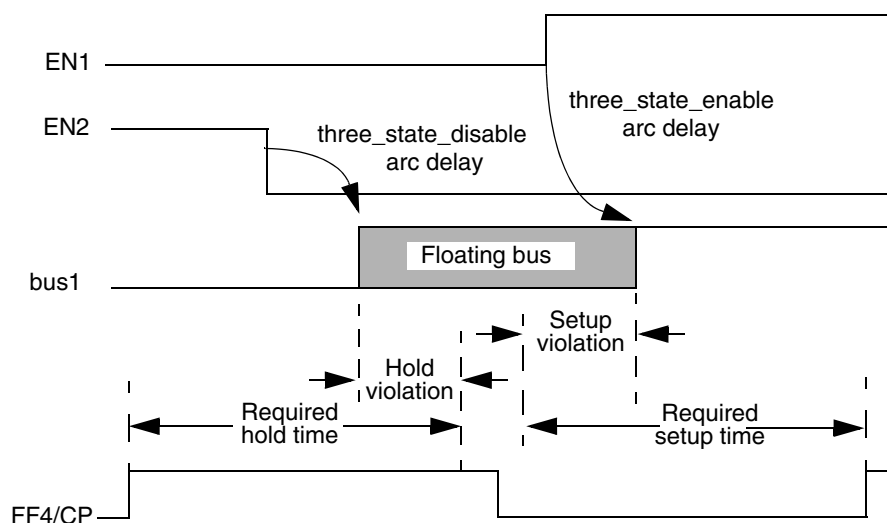
Even when you set this variable to `true`, PrimeTime considers the `three_state_enable` arc delay for setup violation checking and the `three_state_disable` arc delay for hold violation checking. This occurs during a floating bus region, which is described in the next section.

## Performing Floating Bus Checks

A floating bus occurs when the bus is at a valid logic value and then begins to float. Timing analysis ensures that the float does not propagate an X to the register data pin until after the hold check. PrimeTime assumes the same setup and hold relationships as for any other data signals.

Figure 5-18 shows another situation that can arise that yields a transient floating bus condition.

Figure 5-18 Floating Bus Contention Check



In [Figure 5-18](#), the hold violation for FF4/CP is seen only if the `three_state_disable` arc delay is considered. A transition to the Z state must be propagated as a possible transition to logic 0 or logic 1 to find all possible cases. A similar situation happens for the setup check. The setup violation is seen only if the `three_state_enable` arc delay is considered.

Because the `three_state_enable` arc delays are considered, by default PrimeTime checks for all setup and hold violations that occur due to the floating bus region. The floating bus region is bounded by the minimum `three_state_disable` arc delay of any bus driver from one side and by the maximum `three_state_enable` arc delay from the other side. PrimeTime ignores charge decay here (it assumes that the logical value for the bus immediately becomes unknown—(X)—when the bus is floating).

If you know that such floating bus regions can never occur, disable checking for both setup and hold violations that occur due to this bus contention region. To disable the checks, set the `timing_disable_floating_bus_check` variable to `true`. In this case, PrimeTime ignores the `three_state_disable` arc delay for hold violations checking and the `three_state_enable` arc delay for setup violations checking. Even when you set this variable to `true`, the `three_state_enable` arc delay is considered for setup violation checking and the `three_state_disable` arc delay is considered for hold violation checking. This occurs during a bus contention region, which was described in the previous section.

---

## Fast Multidrive Delay Analysis

In large designs with annotated parasitics, the runtime for performing RC delay calculation can be large for massively multidriven networks. The primary bottleneck in this process is the runtime required to compute the trip times for measuring delays and slews at the load pins.

For massively multidriven networks with homogeneous drivers (drivers with similar input skews, slews, and operating conditions), a calculation mode is available that significantly reduces the runtime. This mode shorts all of the driver nodes together with a network of resistors and uses a single driver model at the first driver node for all waveform calculations. The drive strength of the single driver is scaled up to be equivalent to the whole set of original drivers. The delay calculation results for the single driver are copied to all of the other drivers, including delay, slew, driver model parameters, and effective capacitance.

Usage of the fast multidrive analysis mode is based on the `rc_driver_count_threshold_for_fast_multidrive_analysis` variable. This variable is an integer that specifies the maximum number of network drivers that are handled without invoking the fast multidrive analysis mode. The fast mode is invoked when the number of drivers exceeds the variable setting. By default, the variable is set to 9. Setting the variable to 0 disables the mode entirely (the same as setting it to an extremely large number).



When PrimeTime uses the fast multidrive analysis mode, it generates an RC-010 warning message that reports the number of drivers, the number of loads, the input slew and input skew spreads, and the matching or mismatching of driver library timing arcs and driver operating conditions. This information can help determine the accuracy of the analysis.

The accuracy of this mode is best when the driver library arcs are the same and operate under the same context (operating conditions, input slew, input skew, and so on), and operate with the same network impedance. These conditions are often typical of mesh networks. Any differences in operating conditions and input slews can cause small differences in output delay and output slew. For accuracy, the differences in input slews should be small compared to the delays through the network. For maximum accuracy, either disable the fast multidrive analysis mode or annotate delays and slews onto the network.

Note:

For more information, see [“Parallel Driver Reduction” on page 5-41](#) and [“Reducing SDF for Clock Mesh/Spine Networks” in Chapter 8](#).

---

## Parallel Driver Reduction

Because clock signals must typically drive a large number of loads, clocks are often buffered to provide the drive strength necessary to reduce clock skew to an acceptable level. A large clock network might use a large number of drivers operating in parallel. These drivers can be organized in a “mesh” or “spine” pattern to distribute the signal throughout the chip.

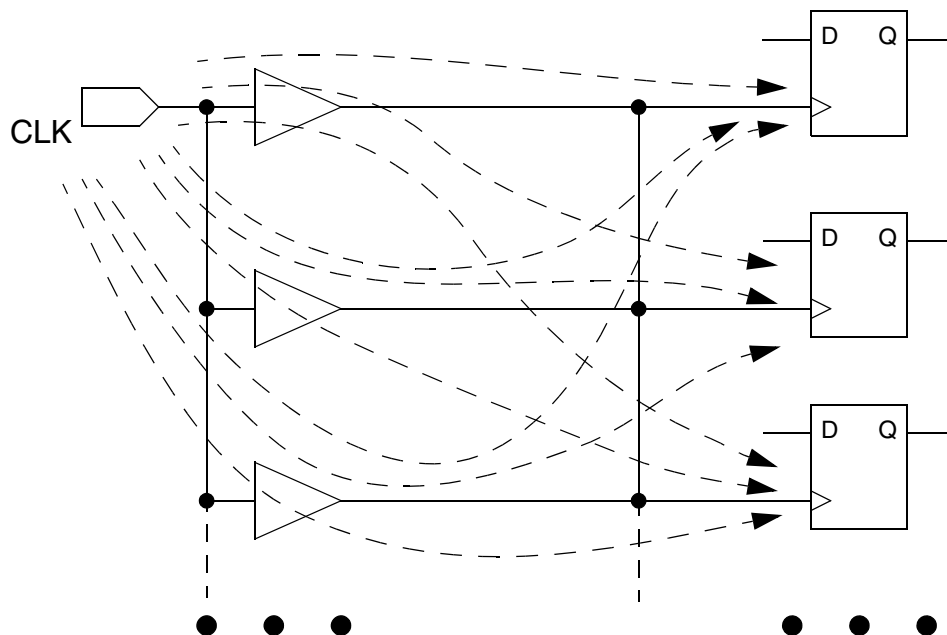
If a design has 1,000 drivers in parallel driving 1,000 loads, PrimeTime must keep track of one million different timing arcs between the drivers and loads. Timing analysis of such a network can consume a large amount of CPU and memory resources.

For better performance, PrimeTime can reduce the number of timing arcs that must be analyzed. The reduction process is illustrated in [Figure 5-19 on page 5-42](#). When the reduction feature is enabled, PrimeTime selects one driver in a parallel network and analyzes the timing arcs through that driver only.

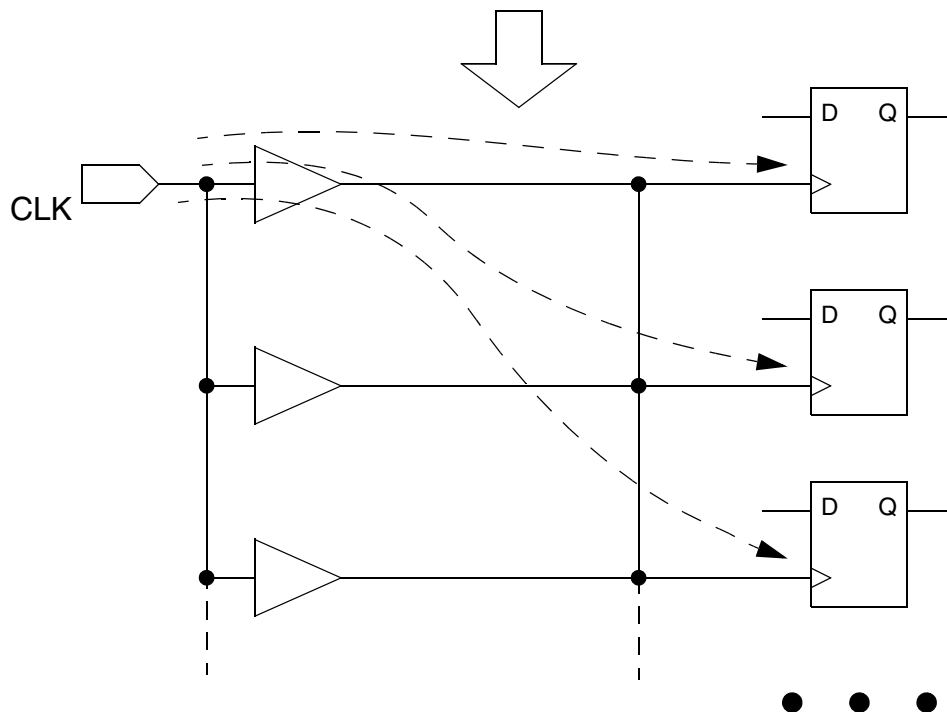
Note:

For related information, see [“Fast Multidrive Delay Analysis” on page 5-40](#) and [“Reducing SDF for Clock Mesh/Spine Networks” in Chapter 8](#).

Figure 5-19 Parallel Driver Reduction



```
pt_shell> set timing_reduce_multi_drive_net_arcs true
pt_shell> set timing_reduce_multi_drive_net_arcs_threshold 1000
pt_shell> link_design
```



---

## Invoking Parallel Driver Reduction

Parallel driver reduction is controlled by the following variables:

- `timing_reduce_multi_drive_net_arcs`
- `timing_reduce_multi_drive_net_arcs_threshold`

The first of these two variables can be set to `true` or `false` to enable or disable parallel driver reduction. By default, it is set to `false` and no reduction is done. When you set the variables to `true` during design linking, PrimeTime checks for the presence of nets with multiple drivers. If it finds a net with driver-load combinations exceeding the threshold specified by the threshold variable, it reduces the number of timing arcs associated with the drivers of that net.

The threshold variable specifies the minimum number of timing arcs that must be present to trigger a reduction. By default, it is set to 10,000, which means that PrimeTime reduces the timing arcs of a net if the number of drivers multiplied by the number of loads on the net is more than 10,000. In typical designs, this large number only occurs in clock networks.

PrimeTime performs driver reduction for a net when all of the following conditions are true:

- Number of driver-load combinations is more than the variable-specified threshold (10,000 by default)
- Driver cells are nonsequential library cells (not flip-flops or latches and not hierarchical)
- All drivers of the net are instances of the same library cell
- All the drivers are connected to the same input and output nets

To expand a reduced network to its original form or to perform driver reduction with a different threshold, you must relink the design.

---

## Working With Reduced Drivers

After layout is complete and detailed parasitic information becomes available, it is not possible to annotate this information on a reduced network. To get accurate results, use an external simulator such as SPICE to get detailed delay information for the network. Then you can annotate the clock latency values and transition times on the individual clock pins of the sequential cells, while still allowing PrimeTime to treat the reduced network as ideal, with zero delay. This technique provides reasonably accurate results, while being very efficient because the clock network timing only needs to be analyzed once, even for multiple PrimeTime analysis runs.

If you back-annotate the design with the `read_sdf` command, any annotations on the reduced timing arcs are ignored. PrimeTime issues a PTE-048 message when this happens. Suppress these messages with the following command:

```
pt_shell> suppress_message PTE-048
```

If you write out SDF with the `write_sdf` command, no interconnect delays are generated for the reduced network.

Reducing parallel drivers only affects the timing arcs analyzed by PrimeTime, not the netlist. Therefore, it does not affect the output of the `write_changes` command. For information about RC delay calculation with massively multidriven networks, see [“Fast Multidrive Delay Analysis” on page 5-40](#).

---

## Data-to-Data Checking

PrimeTime can perform setup and hold checking between two data signals, neither of which is defined to be a clock, at any two pins in the design. This feature can be useful for checking the following types of timing constraints:

- Constraints on handshaking interface logic
- Constraints on asynchronous or self-timed circuit interfaces
- Constraints on signals with unusual clock waveforms that cannot be easily specified with the `create_clock` command
- Constraints on skew between bus lines
- Recovery and removal constraints between asynchronous preset and clear input pins

A timing constraint between two data (nonclock) signals is called a nonsequential constraint. You can define such checks in PrimeTime by using the `set_data_check` command, or define them for a library cell by setting nonsequential constraints for the cell in Library Compiler. You should use data checks only in situations such as those described above. Data checks should not be considered a replacement for standard sequential checking.

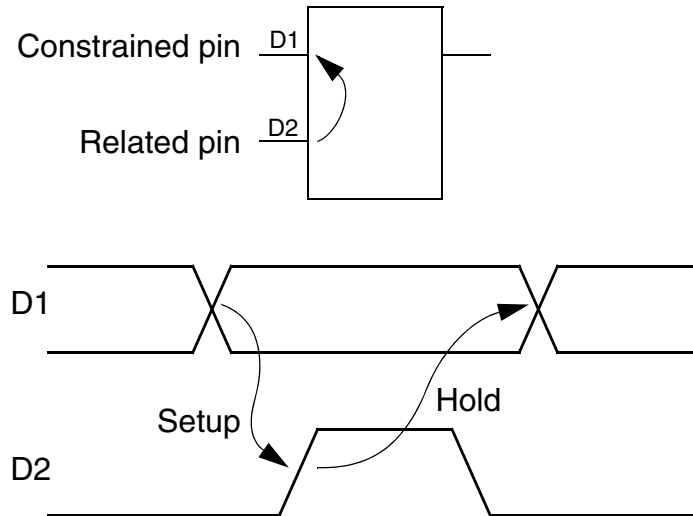
---

## Data Check Examples

[Figure 5-20 on page 5-45](#) shows a simple example of a cell that has a nonsequential constraint. The cell has two data inputs, D1 and D2. The rising edge of D2 is the active edge that might be used to latch data at D1. Pin D1 is called the constrained pin and Pin D2 is

called the related pin. In a sequential setup or hold check, pin D2 would be considered the clock pin. However, for any of a number of reasons, it might be desirable to consider the signal at D2 a data signal, and not define it to be a clock.

*Figure 5-20 Simple Data Check Example*



In this example, the signal at D1 must be stable for a certain setup time before the active edge. It must also be stable for a certain hold time after the active edge. If these nonsequential constraints are not already defined for the library cell, you can define them in PrimeTime. To do so, use commands similar to the following:

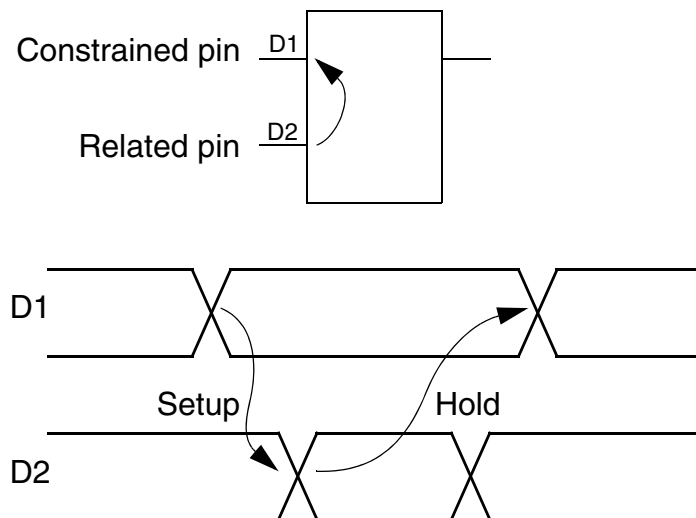
```
pt_shell> set_data_check -rise_from D2 -to D1 -setup 3.5
pt_shell> set_data_check -rise_from D2 -to D1 -hold 6.0
```

The “from” pin is the related pin and the “to” pin is the constrained pin. If the data checks apply to both rising and falling edges on the related pin, use `-from` instead of `-rise_from` or `-fall_from`, as shown in the following example:

```
pt_shell> set_data_check -from D2 -to D1 -setup 3.5
pt_shell> set_data_check -from D2 -to D1 -hold 6.0
```

The resulting timing checks are illustrated in [Figure 5-21](#).

*Figure 5-21 Data Checks on Rising and Falling Edges*

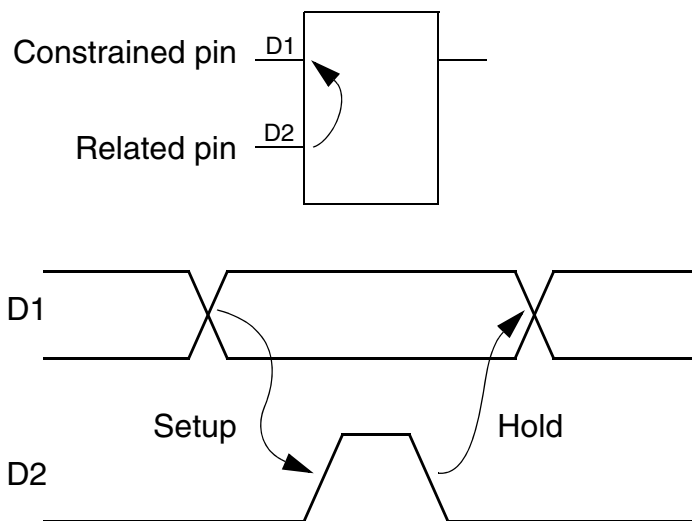


Define a no-change data check by specifying only a setup check from the rising edge and a hold check from the falling edge:

```
pt_shell> set_data_check -rise_from D2 -to D1 -setup 3.5
pt_shell> set_data_check -fall_from D2 -to D1 -hold 3.0
```

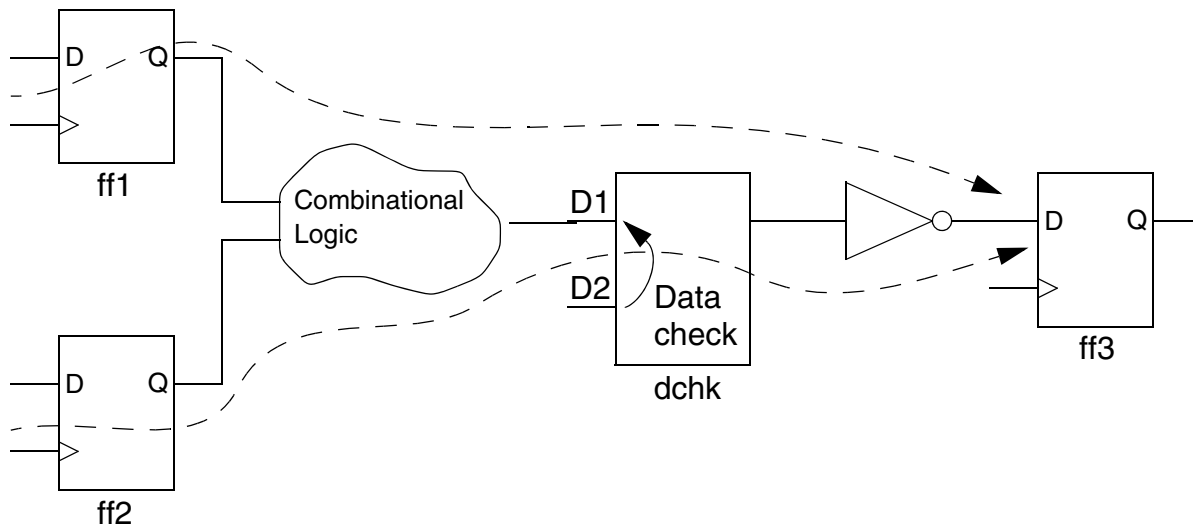
PrimeTime interprets this as a no-change check on a positive-going pulse. The resulting timing check is illustrated in [Figure 5-22](#).

*Figure 5-22 No-Change Data Check*



Data checks are nonsequential, so they do not break timing paths. For example, in [Figure 5-23](#), the data check between D1 and D2 does not interrupt the timing paths shown by the dashed-line arrows. If you were to define the signal at D2 to be a clock, the check would be sequential and the paths would be terminated at D1.

*Figure 5-23 Timing Paths Not Broken by Data Checks*



You can specify a data check pin as a path endpoint for the `report_timing` command. In that case, PrimeTime reports the data checks that apply to the pin. For example, for the circuit shown in [Figure 5-23](#), `report_timing -to dchk/D1` generates a data check report, whereas `report_timing -through dchk/D1` generates a timing report on standard paths that pass through the specified pin.

To remove data checks set with the `set_data_check` command, use the `remove_data_check` command. For more information about either command, see the specific man page.

## Data Checks and Clock Domains

In a data check, signals arriving at a constrained pin or related pin can come from different clock domains. PrimeTime checks the signal paths separately and puts them into different clock groups, just like standard sequential checks.

If the related pin has signals from multiple clock domains, you might want to specify which clock domain to analyze at that pin for the data check. To specify a clock domain to analyze, either use the `-clock clock_name` option of the `set_data_check` command, or disable all clocks other than the clock of interest.

---

## Library-Based Data Checks

PrimeTime performs data checking for any cell that has nonsequential timing constraints defined in the library cell, as long as the signal at the related pin is not defined to be a clock in PrimeTime. If the signal is defined to be a clock, PrimeTime converts the nonsequential checks to sequential checks and does not block this clock signal from further propagation. If a combinational arc from the related pin exists (such as with an integrated clock gating cell), the clock is free to continue propagation down this arc.

In Library Compiler, you define nonsequential constraints on a cell by specifying a related pin and by assigning the following `timing_type` attributes to the constrained pin:

```
non_seq_setup_rising
non_seq_setup_falling
non_seq_hold_rising
non_seq_hold_falling
```

For more information about defining nonsequential constraints in Library Compiler, see the Library Compiler user guides.

Defining nonsequential constraints in the library cell results in a more accurate analysis than using the `set_data_check` command because the setup and hold times can be made sensitive to slew of the constrained pin and the related pin. The `set_data_check` command is not sensitive to slew.

To specify which clock domain to use at the related pin for data checks defined in library cells, use the `set_data_check ... -clock clock_name` command. The `remove_data_check` command does not remove data checks defined in library cells.

---

## Data Propagation Through Generated Clocks

To support the use of multiple levels of undefined generated clocks, use the `timing_propagate_through_unclocked_registers` variable. When this variable is set to `true`, PrimeTime propagates data through a flip-flop or level-sensitive latch when there is no defined clock driving the clock pin. When the variable is `false` (the default), the lack of a defined clock prevents data propagation through a flip-flop or latch.

---

## Interdependent Setup and Hold Pessimism Reduction

Setup and hold pessimism reduction (SHPR) allows you to reduce pessimism in slack computation in path-based analysis. This is done by using the information in a library that supports interdependent setup and hold characterization data.



In a traditional library, the setup and hold constraint arcs for a sequential cell have a single fixed behavior. Setup might be characterized conservatively to allow hold constraint arcs to be characterized aggressively or vice versa. In some libraries, both the setup and hold constraints might be characterized conservatively (large setup/hold requirements) to avoid any possibility of failure during operation. In other libraries, both constraints might be characterized aggressively (small setup/hold requirements) to achieve maximum performance. If both setup and hold are characterized aggressively, failures can result unless care is taken to avoid data pulse widths that are too narrow to be reliably captured by the clock edge.

In reality, the magnitude of the setup and hold constraint requirements depend on each other. In a library with SHPR data, PrimeTime is able to understand this interdependence between setup and hold and trade off the setup and hold checks against each other to adapt to the needs of each sequential cell to the upstream logic during path-based analysis. By using the SHPR data, PrimeTime is able to use aggressive setup/hold constraints while still ensuring the minimum data pulse width requirements for reliable capture are met.

---

## Use Model for SHPR

PrimeTime supports three modes for SHPR as explained in the sections below. Some user-controlled constraints of SHPR optimization are also available.

- [Setup-Preferred Slack Improvement](#)
- [Hold-Preferred Slack Improvement](#)
- [Total Negative Slack Improvements](#)
- [SHPR Optimization Constraints](#)

### Setup-Preferred Slack Improvement

For some designs that prefer fast clock frequency (and assuming the hold time violation is easy to fix), the setup-preferred slack improvement mode can be desirable. This mode optimizes only total negative setup slacks of `max_rise` and `max_fall` paths ending at a specific flip-flop, with the trade off of corresponding hold slacks.

### Hold-Preferred Slack Improvement

For some designs in which the setup time violation can be easily fixed by lowering clock frequency, hold time violation fixing becomes the primary consideration. The hold-preferred slack improvement mode is designed for such a case. This mode optimizes only total negative hold slacks of `min_rise` and `min_fall` paths ending at a specific flip-flop, with the trade off of corresponding setup slacks.

## Total Negative Slack Improvements

The total negative slack mode minimizes the sum of negative slacks of all four types of timing paths ending at a particular flip-flop. By default, this mode allows trade off between the positive slack of one type of timing path and the negative slack of another type of timing path, without incurring new timing violations. This mode gives the same priorities on both negative setup slack and negative hold slack. This is the default mode of SHPR.

## SHPR Optimization Constraints

For the three modes just described, user controls are provided to influence the slack trade-off. This is known as the SHPR optimization constraint, which defines the limit of trade off of the positive setup or hold slack. When one type of slack is sacrificed to improve the other type of slack, specify the slack limit for the sacrificial slack trade off.

---

## SHPR Optimization Mechanism

In SHPR, a sequential capturing endpoint can be traded off in only two independent scenarios:

- `setup_rise` against `hold_fall`
- `setup_fall` against `hold_rise`

[Figure 5-24](#) depicts these two scenarios.

*Figure 5-24 Two Setup and Hold Trade off Scenarios*

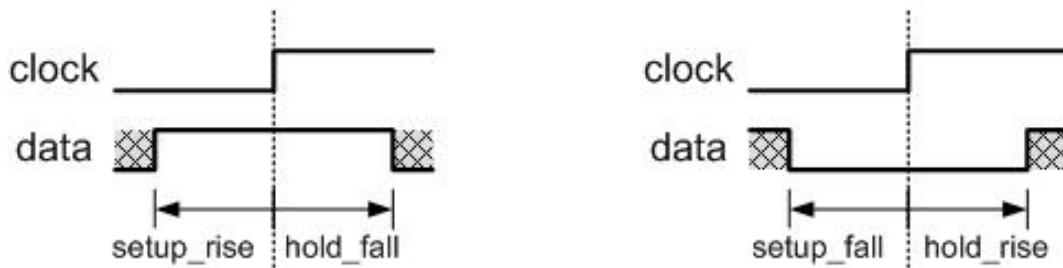
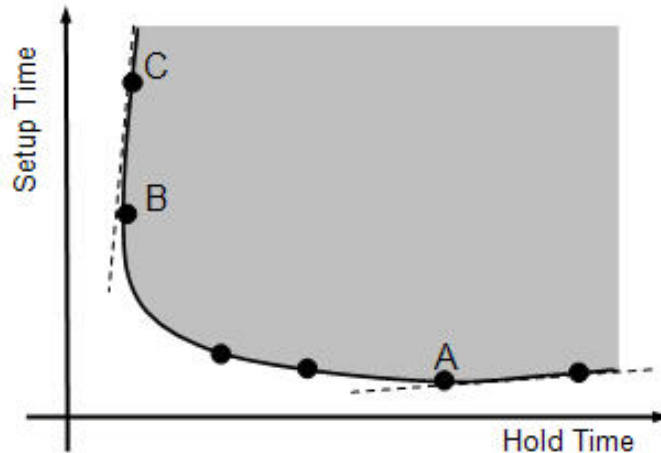


Figure 5-25 shows a typical interdependent setup and hold value curve. On the curve, any points beyond point A or B have a positive slope. Points A and B are called the turning points.

Figure 5-25 Interdependent Setup and Hold Value Curve



To be conservative, PrimeTime does not allow any extrapolation beyond the boundary points of the characterized SHPR curve.

## SHPR User Interface

To set the optimization constraints for SHPR, use the `set_setup_hold_pessimism_reduction` command. The valid modes are total, setup, and hold. Use the different modes to control how setup slack is traded off for hold slack or vice versa.

In setup mode (also known as setup-preferred mode), the goal is to prefer positive setup slack at the expense of hold slack. In hold mode (also known as hold-preferred mode), the goal is to prefer positive hold slack at the expense of setup slack. In total mode (the default), the goal is to trade off setup and hold slack in a way that minimizes the sum of the negative rise/fall and setup/hold slack. This mode results in the smallest overall set of setup and hold timing violations.

It might not be desirable to introduce a large violation in the sacrificial slack type to gain only a small improvement in the preferred slack type. Use the `-setup_cutoff` and `-hold_cutoff` options of the `set_setup_hold_pessimism_reduction` command to keep the trade off reasonable. In setup-preferred mode, hold violations are made no worse than the `-hold_cutoff` value to improve setup. In hold-preferred mode, setup violations are made no worse than the `-setup_cutoff` value to improve hold. The default value for both cutoff options is negative infinity, which allows any amount of slack worsening in the sacrificial slack to improve the preferred slack.

In both setup and hold mode, the preferred slack type is improved only as far as it takes to reach zero slack and no further.

To disable SHPR, use the `remove_setup_hold_pessimism_reduction` command, which has the following options:

- `-setup_cutoff`: reset cutoff setup slack
- `-hold_cutoff`: reset cutoff hold slack

If you do not add an option, the SHPR feature is completely disabled.

---

## SHPR Examples

The following sections outline some examples of SHPR.

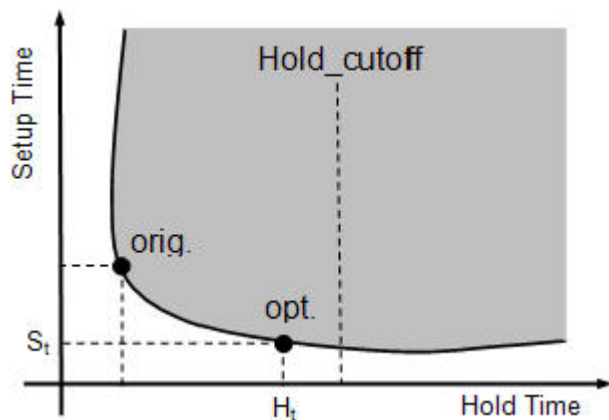
### Setup Preferred Slack Improvement Example

In the following example, the mode is setup preferred slack improvement and the cutoff hold slack is -100 ps.

```
pt_shell>set_setup_hold_pessimism_reduction -mode setup \
 -hold_cutoff -100
```

The targeted setup time that results in zero slack in the max timing path is determined as  $S_t$  in [Figure 5-26](#) below.

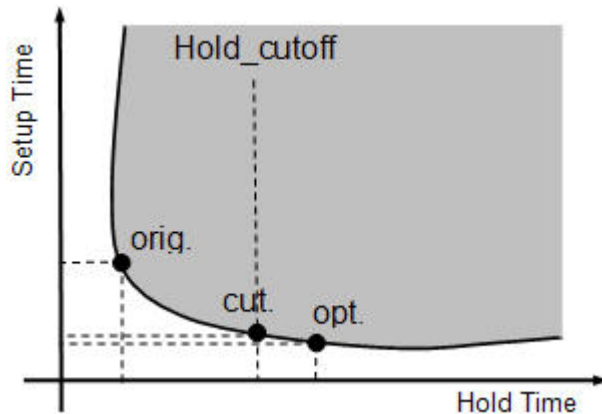
*Figure 5-26 Successful Setup Slack Improvement*



Using  $S_t$  as the input, obtain the corresponding hold time  $H_t$ , by using the interdependent setup and hold value curve. If  $H_t$  does not reach the `hold_cutoff` value,  $S_t$  and  $H_t$  are the optimized values of setup and hold time, as shown in [Figure 5-26](#).

Instead, if  $H_t$  is beyond the `hold_cutoff` value, as shown below in [Figure 5-27](#), the corresponding setup and hold times of the cutoff point are considered as the optimized setup and hold values.

Figure 5-27 Setup Slack Improvement Stops in Cutoff Hold Slack



## Total Slack Improvement Example

The total mode is a special case. It does not find the minimal negative slack of min and max timing paths along the whole interdependent setup and hold curve. Instead, one of the following two cases with the minimal total negative slack of min and max timing paths is chosen:

- Case 1: [setup-preferred slack improvement mode, `hold_cutoff`]
- Case 2: [hold-preferred slack improvement mode, `setup_cutoff`]

## Liberty Format Extension

The Liberty format and Library Compiler support interdependent setup and hold tables, starting in the Y-2006.06 release. The `interdependence_id` tag is used to identify the interdependent setup and hold tables. [Figure 5-28](#) shows the library format needed to support SHPR.

Figure 5-28 Example of Library Format Needed to Support SHPR

|                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> timing() {   related_pin : "CP";   timing_type : setup_rising;   rise_constraint(setup_template_3x3) {     index_1 ("0.0264, 0.1728, 0.6736");     index_2 ("0.0264, 0.1728, 1.3424");   }   values("xxxx, xxxx, xxxx", \          "xxxx, xxxx, xxxx", \          "xxxx, xxxx, xxxx"); } fall_constraint(setup_template_3x3) {   index_1 ("0.0264, 0.1728, 0.6736");   index_2 ("0.0264, 0.1728, 1.3424"); } values("xxxx, xxxx, xxxx", \        "xxxx, xxxx, xxxx", \        "xxxx, xxxx, xxxx"); } } timing() {   related_pin : "CP";   timing_type : setup_rising;   interdependence_id : 1;   rise_constraint(setup_template_3x3) {     index_1 ("0.0264, 0.1728, 0.6736");     index_2 ("0.0264, 0.1728, 1.3424");   }   values("xxxx, xxxx, xxxx", \          "xxxx, xxxx, xxxx", \          "xxxx, xxxx, xxxx"); } fall_constraint(setup_template_3x3) {   index_1 ("0.0264, 0.1728, 0.6736");   index_2 ("0.0264, 0.1728, 1.3424"); } values("xxxx, xxxx, xxxx", \        "xxxx, xxxx, xxxx", \        "xxxx, xxxx, xxxx"); } } </pre> | <pre> timing() {   related_pin : "CP";   timing_type : hold_rising;   rise_constraint(hold_template_3x3) {     index_1 ("0.0264, 0.1728, 0.6736");     index_2 ("0.0264, 0.1728, 1.3424");   }   values("yyyy, yyyy, yyyy", \          "yyyy, yyyy, yyyy", \          "yyyy, yyyy, yyyy"); } fall_constraint(hold_template_3x3) {   index_1 ("0.0264, 0.1728, 0.6736");   index_2 ("0.0264, 0.1728, 1.3424"); } values("yyyy, yyyy, yyyy", \        "yyyy, yyyy, yyyy", \        "yyyy, yyyy, yyyy"); } } timing() {   related_pin : "CP";   timing_type : hold_rising;   interdependence_id : 1;   rise_constraint(hold_template_3x3) {     index_1 ("0.0264, 0.1728, 0.6736");     index_2 ("0.0264, 0.1728, 1.3424");   }   values("yyyy, yyyy, yyyy", \          "yyyy, yyyy, yyyy", \          "yyyy, yyyy, yyyy"); } fall_constraint(hold_template_3x3) {   index_1 ("0.0264, 0.1728, 0.6736");   index_2 ("0.0264, 0.1728, 1.3424"); } values("yyyy, yyyy, yyyy", \        "yyyy, yyyy, yyyy", \        "yyyy, yyyy, yyyy"); } } </pre> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

For more information, see the *Library Compiler Technology and Symbol Libraries Reference Manual*.

# 6

## Advanced On-Chip Variation (AOCV)

---

Advanced on-chip variation (AOCV) is a technology that recognizes the need for closer integration between design methodology and fabrication process variation. This technology reduces unnecessary pessimism by analyzing the physical location and logic depth of paths in a design. This chapter explains the graph-based and path-based AOCV solutions and provides you with flow and reporting information.

This chapter contains the following sections:

- [Introduction](#)
- [AOCV Flow](#)
- [Specifying the Scope of the AOCV Analysis](#)
- [Importing AOCV Information](#)

---

## Introduction

AOCV is an optional accuracy improvement that determines derating factors based on metrics of path logic depth and the physical distance traversed by a particular path. A longer path that has more gates tends to have less total variation because the random variations from gate-to-gate tend to cancel each other out. A path that spans a larger physical distance across the chip tends to have larger systematic variations. AOCV is less pessimistic than a traditional OCV analysis, which relies on constant derate factors that do not take path-specific metrics into account.

The AOCV analysis determines path-depth and location-based bounding box metrics to calculate a context-specific AOCV derate factor to apply to a path, replacing the use of a constant derate factor.

The AOCV solution works with all other PrimeTime features and affects all reporting commands. This solution works in both the Standard Delay Format (SDF)-based and the delay calculation based flows. It is compatible with distributed multi-scenario analysis and multicore analysis.

---

## AOCV Flow

The AOCV solution in PrimeTime is integrated into the graph-based and path-based analysis capabilities, which enables a successive refinement strategy of analysis that is already used today with traditional static timing analysis in PrimeTime. In AOCV flows, sign-off is performed using graph-based AOCV, with path-based AOCV as an optional capability to refine the analysis of critical paths. Graph-based and path-based analysis with AOCV is analogous to graph-based and path-based analysis in the traditional static timing analysis context.

---

### Graph-Based AOCV Solution

Graph-based AOCV analysis is a fast, design-wide analysis performed during the `update_timing` command. It allows designers to exploit reduced derating pessimism across the entire design to reduce silicon area and improve design performance. You set the `timing_aocvm_enable_analysis` variable to `true` to enable the graph-based AOCV analysis.

By construction, graph-based analysis always provides a conservative analysis compared to path-based analysis to prevent an optimistic calculation. With graph-based AOCV analysis, PrimeTime computes conservative values of depth and distance metrics. To perform graph-based AOCV analysis, conservative values of path-depth and location-based bounding box



are chosen to bound the worst-case path through a cell. For example, in graph-based AOCV analysis, the depth count for a cell does not exceed that of the path of minimum depth that traverses that cell. Otherwise, the analysis might be optimistic for the minimum-depth path.

Graph-based AOCV is a prerequisite for further margin reduction using path-based AOCV analysis on selected critical paths. For most designs, graph-based AOCV is sufficient for sign-off.

---

## Path-Based AOCV Solution

The AOCV metrics computation is different for graph-based and path-based AOCV. In graph-based AOCV, the depth and distance metrics are calculated for all paths through each timing arc. In path-based mode, the AOCV metrics are calculated precisely for each timing path.

To further reduce pessimism and improve accuracy of AOCV results, use the `report_timing` command with the `-pba_mode` option to analyze paths in isolation from other paths. For example,

```
pt_shell> report_timing -pba_mode path
```

The PrimeTime path-based analysis engine performs both AOCV path-based analysis and regular path-based analysis by default. For the AOCV flow, it is strongly recommended that you set the `pba_aocvm_only_mode` variable to `true` so that only AOCV path-based analysis is applied. For more information about path-based analysis, see [“Path-Based Timing Analysis” on page 5-14](#).

Although it is strongly advised that AOCV path-based analysis usage should always follow an AOCV graph-based analysis, AOCV path-based analysis within a traditional OCV flow is also permitted. When you have not enabled AOCV graph-based analysis, the derate factors computed by PrimeTime during AOCV path-based analysis are bounded by constant derates specified using the `set_timing_derate` command. The specification of these constant timing derate factors is required for this flow and it is your responsibility to ensure they are conservative.

---

## Specifying the Scope of the AOCV Analysis

The AOCV feature is enabled by setting the `timing_aocvm_enable_analysis` variable to `true`. You can then configure an AOCV analysis using the `timing_aocvm_analysis_mode` variable. You can include the following analysis modes in this variable:

- `clock_network_only`

When you specify this option, the derivation and application of AOCV derates is restricted to arcs that are in the clock network. Clock network AOCV depth and distance metrics are calculated, considering only clock network topology. If constant derates have been annotated for data network objects, the data network receives constant on-chip variation (OCV) derating. Otherwise, the data network is not derated.

- `combined_launch_capture_depth`

Separate launch and capture depths are calculated in AOCV by default. By specifying this option, launch and capture paths are considered together and a combined depth is calculated for the entire path.

- `separate_data_and_clock_metrics`

When you specify this option, the depths of data and clock paths are computed separately. The respective computed depths are used to separately determine the applicable derate values for clock paths and data paths. This option cannot be used along with the `combined_launch_capture_depth` option.

- `single_path_metrics`

When you specify this option, single path metrics are calculated for all path objects. This behavior is backwardly compatible with the Tcl-based location aware on-chip variation (LOCV) solution.

For more information about the analysis modes, see the `timing_aocvm_analysis_mode` and `timing_aocvm_analysis_mode` variables man page.

---

## Importing AOCV Information

PrimeTime allows you to specify AOCV information using derate tables, which are required for an AOCV analysis. Guard-band timing derates allow you to optionally model nonprocess-related effects in an AOCV flow. Importing AOCV information is described in more detail in the following sections:

- [Specifying Derate Tables](#)
- [File Format for AOCV](#)

- [Specifying Random Coefficients](#)
- [Guard-Banding in AOCV](#)
- [Incremental Timing Derates](#)
- [Using OCV Derates in AOCV Analysis](#)

---

## Specifying Derate Tables

Use the `read_aocvm` command to read AOCV derate tables from a disk file. Derate tables are annotated onto one or more design objects and are applied directly to timing arc delays. The allowed design object classes are hierarchical cells, library cells, and designs. For more information about the `read_aocvm` command, see the man page.

Use the `write_binary_aocvm` command to create binary encoded AOCV files from ASCII-AOCV files. This command is used to protect sensitive process-related information. The `read_aocvm` command can read binary and compressed binary AOCV files that were created using the `write_binary_aocvm` command. No additional arguments are required to read binary or compressed binary AOCV files.

Internally calculated AOCV derates are shown in the derate column of the `report_timing` command if you have specified the `-derate` option.

To display AOCV derate table data, use the `report_aocvm` command. An AOCV derate table imported from a binary or compressed binary file is not visible in the `report_aocvm` output. You can show design objects annotated with early, late, rise, fall, cell, or net derate tables. You can also use this command to determine cells and nets that have been annotated or not annotated with AOCV information.

In a graph-based AOCV analysis, specify a timing arc in the `object_list` of the `report_aocvm` command. The graph-based depth and distance metrics and AOCV derate factors on timing arc objects are displayed.

```
pt_shell> report_aocvm [get_timing_arcs -of_objects [get_cells U1]]
```

If you specify a timing path in the `object_list`, path metrics (distance, launch depth, and capture depth) for that path are displayed.

```
pt_shell> report_aocvm [get_timing_paths -path_type full_clock_expanded \
 -pba_mode path]
```

For more information about the `report_aocvm` command, see the man page.

## File Format for AOCV

The AOCV file format allows you to specify multiple AOCV derate tables. It supports the following table types:

- One-dimensional tables in either depth or distance
- Two-dimensional tables in both depth and distance

The file format is defined so that you can associate an AOCV table with a group of objects. The objects are selected internally within PrimeTime based using the following definitions:

- `object_type`    `design | lib_cell | cell`
- `object_spec`    `[patterns]`

In the `object_spec` definition, the `patterns` field is optional. It specifies the object name and an expression that you want to evaluate based on the attributes of the object.

You can use regular expression matching for the `patterns` field of the `object_spec` definition. It is the same as the `regexp` Tcl command that you can use for design object gathering commands, such as the `get_cells`, `get_lib_cells`, and `get_designs` commands, in PrimeTime.

You can use any of the options of the related object-gathering command in the `patterns` field. For example, if the `object_type` is `lib_cell`, use any of the arguments of the related `get_lib_cells` command in the `patterns` field. For more information about these commands, see the specific man pages.

PrimeTime can annotate cell and net tables on the design using the `object_type` `design`. It can also annotate cell tables on library cells using the `lib_cell` object class and annotate cell and net tables on hierarchical cells using the `cell` object class.

### Note:

All field descriptions are required, unless otherwise stated. To add a comment in any location within the file, use double forward slashes (`//`).

[Table 6-1](#) shows the syntax definition for the AOCV file format.

*Table 6-1 AOCV File Format Syntax*

| Field specifier          | Definition  | Field description                     |
|--------------------------|-------------|---------------------------------------|
| <code>version</code>     | VERSION     | AOCV version number.                  |
| <code>object_type</code> | OBJECT_TYPE | <code>design   lib_cell   cell</code> |
| <code>rf_type</code>     | RF_TYPE     | <code>rise   fall   rise fall</code>  |

Table 6-1 AOCV File Format Syntax (Continued)

| Field specifier | Definition    | Field description                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-----------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| delay_type      | DELAY_TYPE    | cell   net   cell net                                                                                                                                                                                                                                                                                                                                                                                                                         |
| derate_type     | DERATE_TYPE   | early   late                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| path_type       | PATH_TYPE     | clock   data   clock data                                                                                                                                                                                                                                                                                                                                                                                                                     |
| object_spec     | [patterns]    | string                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| voltage         | voltage_value | float<br>This is an optional field of data type <code>float</code> and the units are in Volts.                                                                                                                                                                                                                                                                                                                                                |
| depth           | [DEPTH]       | A set of M floats, where M can be zero.                                                                                                                                                                                                                                                                                                                                                                                                       |
| distance        | [DISTANCE]    | A set of N floats, where N can be zero.<br>Note:<br>You should use nanometers (nm) for the distance field coordinates in the AOCV tables.                                                                                                                                                                                                                                                                                                     |
| table           | [TABLE]       | A set of N x M floats. There are also the following special cases: <ul style="list-style-type: none"> <li>• If N==0, the table is of size M.</li> <li>• If M==0, the table is of size N.</li> <li>• If M==0 and N==0, the table is of size 1.</li> </ul> Linear interpolation is used to determine points that have not been defined in the table. PrimeTime does not extrapolate beyond the lowest or highest values specified in the table. |

When different `object_type` entries with the same `rf_type` and `derate_type` apply to the same cell or net object, the precedence rules that are used are consistent with the `set_timing_derate` command.

Cell arc derating uses the following precedence, from highest to lowest:

1. library cell
2. hierarchical cell
3. design

Net arc derating uses the following precedence, from highest to lowest:

1. hierarchical cell
2. design

Note:

When multiple table entries with the same `object_type`, `rf_type`, and `derate_type` specifications apply to the same cell or net object, the last table entry takes precedence.

If you specify the `voltage` field, but an associate float value is missing or if you do not specify the voltage field at all, the derate table applies to all voltages. For a multirail cell, it is expected that you specify the derates with either an empty or no voltage field, where these derates are the most conservative ones across all possible input-output voltage combinations of the multirail cell.

The following example of an AOCV file sets an early AOCV table for the whole design, which applies to all cell and nets:

```
version: 1.0
object_type: design
rf_type: rise fall
delay_type: cell net
derate_type: early
object_spec: top
depth: 0 1 2 3
distance: 100 200
table: 0.87 0.93 0.95 0.96 \
 0.83 0.85 0.87 0.90
```

The following example of an AOCV file sets an early AOCV table for the whole design, which applies to all cells with the voltage of 1.2 volts:

```
version: 1.0
object_type: design
rf_type: rise fall
delay_type: cell
derate_type: early
object_spec: top
voltage: 1.2
depth: 0 1 2 3
distance: 100 200
table: 0.88 0.94 0.96 0.97 \
 0.84 0.86 0.88 0.91
```

The following example of an AOCV file sets an early AOCV table for the whole design, which applies to all nets:

```
version 1.0
object_type: design
rf_type: rise fall
delay_type: net
derate_type: early
object_spec: top
depth: 0 1 2 3
distance: 100 200
table: 0.88 0.94 0.96 0.97 \
 0.84 0.86 0.88 0.91
```

The following example of an AOCV file includes the optional `path_type` statement. PrimeTime applies the table data to only the specified path types. To enable separate derating for clock paths and data paths, the `timing_aocvm_analysis_mode` variable must be set to the `separate_data_and_clock_metrics` mode. If the `path_type` statement is omitted, the table applies to both clock paths and data paths. If this statement is used, the version number specified by the `version` statement must be set to 2.0. For example,

```
version: 2.0
object_type: lib_cell
object_spec: LIB/BUF1X
rf_type: rise fall
delay_type: cell
derate_type: late
path_type: data
depth: 1 2 3 4 5
distance: 500 1000 1500 2000
table: \
1.123 1.090 1.075 1.067 1.062 \
1.124 1.091 1.076 1.068 1.063 \
1.125 1.092 1.077 1.070 1.065 \
1.126 1.094 1.079 1.072 1.067
```

---

## Specifying Random Coefficients

Random coefficients can be used to calculate cell path depths. A complex cell can consist internally of a larger number of transistors, so it might be necessary to associate that cell with a logic depth count that is larger than 1 (the default logic depth count for a cell arc). For instance, a buffer is often implemented as two inverters in series. In that case, the buffer cell could be assigned a random derate coefficient of 2.0 to set the logic depth count.

Use the `set_aocvm_coefficient` command to set random AOCV coefficients on cells, library cells, and library timing arcs. Random AOCV coefficients are not required for an AOCV analysis.

```
pt_shell> set_aocvm_coefficient 2.0 [get_lib_cells lib1/BUF2]
```

---

## Guard-Banding in AOCV

Guard-band timing derate allows you to model nonprocess-related effects in an AOCV flow. The `-aocvm_guardband` option is available in the `set_timing_derate`, `report_timing_derate`, and `reset_timing_derate` commands.

Use the `set_timing_derate` command to specify a guard-band derate factor. The `-aocvm_guardband` option is applicable only in an AOCV context. The derate factor that is applied to an arc is a product of the guard-band derate and the AOCV derate. The guard-band derate has no impact outside the context of AOCV.

To report timing derate factors on the current design, use the `report_timing_derate` command. To report only guard-band derate factors, specify the `-aocvm_guardband` option of the `report_timing_derate` command. If you do not specify either the `-variation` or `-aocvm_guardband` option, only the deterministic derate factors are reported. These two options are mutually exclusive.

To reset guard-band derate factors, use the `-aocvm_guardband` option of the `reset_timing_derate` command.

For more information about these commands, see the man pages.

---

## Incremental Timing Derates

Incremental timing derate enables you to fine-tune the timing derate on your objects, such as cells or nets. Use the `-increment` option of the `set_timing_derate` command to specify that the derate value is an incremental timing derate. You cannot incrementally apply guard-band timing derates; therefore, the `-increment` and `-aocvm_guardband` options of the `set_timing_derate` command are mutually exclusive.

Incremental timing derates adhere to the same precedence and override rules as regular timing derates. For more information about the precedence rules, see [“File Format for AOCV” on page 6-6](#).

A single incremental derate value per object, such as a cell or net, is calculated. If multiple incremental derates are applied to the same object, the last value overrides all other values. If no timing derate value exists, the value of 0.0 is used for incremental derates and 1.0 is



used for regular derates. The incremental timing derate is added to the regular timing derate value. If the final resulting derate value is less than 0.0, the negative derate factor is converted to 0.0 to avoid negative delays, and an error message is issued.

Here are some examples of how incremental timing derates are calculated.

- In an AOCV analysis, you might have a u1/u252 cell that has a regular late derate of 1.082 and early derate of 0.924.

```
set_timing_aocvm_enable_analysis true
set_timing_derate -increment -late 0.03 [get_cells u1/u252]
set_timing_derate -increment -early -0.03 [get_cells u1/u252]
```

The final late derate on cell u1/u252 =  $1.082 + 0.03 = 1.112$ , and the final early derate on cell u1/u252 =  $0.924 + (-0.03) = 0.894$ .

- In an OCV analysis, you might have the following values:

```
set_timing_derate -late 1.09
set_timing_derate -early 0.91
set_timing_derate -late 1.11 [get_cells u1/u252]
set_timing_derate -increment -late 0.04 [get_cells u1/u252]
set_timing_derate -increment -early -0.02 [get_cells u1/u252]
set_timing_derate -increment -early -0.03 [get_cells u1/u252]
```

The final late derate on cell u1/u252 =  $1.11 + 0.04 = 1.15$ , and the final early derate on cell u1/u252 =  $0.91 + (-0.03) = 0.88$ .

To reset only incremental derate factors for the specified object, use the `-increment` option of the `reset_timing_derates` command.

To report incremental timing derate factors, use the `-increment` option of the `report_timing_derate` command. In the following example, incremental derate factors have been set globally on the design; therefore, only global incremental derate factors are reported.

```
pt_shell> set_timing_derate -data -increment -early -0.02
pt_shell> set_timing_derate -data -increment -late 0.06
pt_shell> set_timing_derate -clock -increment -early -0.05
pt_shell> set_timing_derate -clock -increment -late 0.10
pt_shell> report_timing_derate -increment
```

```

```

```
Report : timing derate
 -scalar -increment
```

```
Design : simple_path
```

```

```

|                     | ----- Clock ----- |      |       |      | ----- Data ----- |      |       |      |
|---------------------|-------------------|------|-------|------|------------------|------|-------|------|
|                     | Rise              |      | Fall  |      | Rise             |      | Fall  |      |
|                     | Early             | Late | Early | Late | Early            | Late | Early | Late |
| design: simple_path |                   |      |       |      |                  |      |       |      |
| Net delay static    | -0.05             | 0.10 | -0.05 | 0.10 | -0.02            | 0.06 | -0.02 | 0.06 |

|                   |       |      |       |      |       |      |       |      |
|-------------------|-------|------|-------|------|-------|------|-------|------|
| Net delay dynamic | -0.05 | 0.10 | -0.05 | 0.10 | -0.02 | 0.06 | -0.02 | 0.06 |
| Cell delay        | -0.05 | 0.10 | -0.05 | 0.10 | -0.02 | 0.06 | -0.02 | 0.06 |
| Cell check        | --    | --   | --    | --   | --    | --   | --    | --   |

---

## Using OCV Derates in AOCV Analysis

PrimeTime supports a unified framework so that AOCV can use OCV derates under certain conditions. Enable the graph-based AOCV analysis by setting the `timing_aocvm_enable_analysis` variable to `true`. Use the `timing_aocvm_ocv_precedence_compatibility` variable to control the fallback to OCV derates when AOCV is enabled.

When the `timing_aocvm_ocv_precedence_compatibility` variable is set to its default of `true`, the OCV derates are completely ignored in an AOCV analysis. The existing precedence rules specified in the `set_timing_derate` command apply. Therefore, the cell arc derating uses the following order of precedence, from highest to lowest priority:

1. AOCV library cell
2. AOCV hierarchical cell
3. AOCV design

When you set the `timing_aocvm_ocv_precedence_compatibility` variable to `false`, both OCV and AOCV derates are considered for a given object. If derate values are at the same level, the AOCV value is selected over the OCV value. To ensure the correct derate value is applied for a given design object when AOCV is enabled, cell arc derating uses the following order of precedence, from highest to lowest priority:

1. OCV leaf-level cell
2. AOCV library cell
3. OCV library cell
4. AOCV hierarchical cell
5. OCV hierarchical cell
6. AOCV design
7. OCV design

For more information about OCV analysis, see the *PrimeTime Fundamentals User Guide*.

# 7

## Ideal Network Support

---

PrimeTime allows for the creation of ideal networks, on which no design rule checking (DRC) constraints run. It is desirable that designers are able to ignore networks (large unoptimized networks with high fanout and capacitance) in pre-layout and focus instead on violations arising from other sources. Using ideal networks reduces runtime because PrimeTime uses “ideal timing” rather than internally calculated timing. In this way, ideal networks are similar to ideal clock networks, but they can also be applied to data networks.

The setting of an ideal network is described in the following sections:

- [Introduction to Ideal Networks](#)
- [Propagating Ideal Network Properties](#)
- [Using Ideal Networks](#)
- [Using Ideal Latency](#)
- [Using Ideal Transition](#)

---

## Introduction to Ideal Networks

PrimeTime provides the ability to indicate ideal networks - sets of connected ports, pins, nets, and cells - and have them exempt from timing updates and DRC constraint fixing. That is, ideal networks ignore `max_capacitance`, `max_fanout`, and `max_transition` design rule checks. When you specify the source of the ideal network, the pins, ports, nets, and cells contained therein are treated as ideal objects. You or ideal propagation must mark ideal objects.

---

## Propagating Ideal Network Properties

When you specify the source objects (ports, leaf-level pins) of an ideal network, the nets, cells, and pins in the transitive fanout of the source objects can be treated as ideal.

Propagation of the ideal network property is governed by the following rules:

- Pin is marked as ideal if it is one of the following:
  - Pin is specified in the object list of the `set_ideal_network` command
  - Driver pin and its cell are ideal
  - Load pin attached to an ideal net
- Net is marked as ideal if all its driving pins are ideal.
- Combinational cell is marked as ideal if either all input pins are ideal or it is attached to a constant net and all other input pins are ideal.

**Note:**

Ideal network propagation can traverse combinational cells, but it stops at sequential cells.

PrimeTime propagates the ideal network during a timing update and propagates again from ideal source objects as necessary to account for changes in the design, for example, ECOs.

---

## Using Ideal Networks

Using the `set_ideal_network` command, specify which networks you want set as ideal. Indicate the sources of the ideal network with the `object_list` argument. For example, the following command sets the ideal network property on the input port P1, which is propagated along the nets, cells, and pins in the transitive fanout of P1:

```
pt_shell> set_ideal_network P1
```

Use the `-no_propagate` option to limit the propagation of the ideal network for only the nets and pins that are electrically connected to the ideal network source. If, for example, you enter

```
pt_shell> set_ideal_network -no propagate net1
```

only the net, net1, its driver pins, and its load pins are marked as ideal. No further propagation is performed.

If you do not use the `-no_propagate` option with this command, PrimeTime sets the ideal property on the specified list of pins, ports, or nets and propagates this property according to the propagation rules defined in the previous section.

You can remove the ideal network you set by using the `remove_ideal_network` command. To generate a report querying the port, pins, nets, or cells in the design to check whether or not they are ideal, use the `report_ideal_network` command. The following report shows the port, pin, nets, and cells identified as being part of an ideal network:

```
pt_shell> report_ideal_network -timing -load_pin -net -cell
```

```

Report : ideal_network
 -timing
 -load_pin
 -net
 -cell
Design : middle
Version: Y-2006.06-DEV
Date : Tue Feb 21 03:30:19 2006

```

| Source ports<br>and pins | Latency |       |       |       | Transition |       |       |       |
|--------------------------|---------|-------|-------|-------|------------|-------|-------|-------|
|                          | Rise    |       | Fall  |       | Rise       |       | Fall  |       |
|                          | min     | max   | min   | max   | min        | max   | min   | max   |
| -----                    | -----   | ----- | ----- | ----- | -----      | ----- | ----- | ----- |
| middle/p_in              | --      | --    | --    | --    | --         | --    | --    | --    |

| Internal pins<br>with ideal timing | Latency |       |       |       | Transition |       |       |       |
|------------------------------------|---------|-------|-------|-------|------------|-------|-------|-------|
|                                    | Rise    |       | Fall  |       | Rise       |       | Fall  |       |
|                                    | min     | max   | min   | max   | min        | max   | min   | max   |
| -----                              | -----   | ----- | ----- | ----- | -----      | ----- | ----- | ----- |
| n3_i/B                             | 5.62    | 5.62  | --    | --    | --         | --    | --    | --    |
| n0_i/Z                             | 1.34    | --    | 1.34  | --    | --         | --    | --    | --    |

| Boundary pins | Latency |       |       |       | Transition |       |       |       |
|---------------|---------|-------|-------|-------|------------|-------|-------|-------|
|               | Rise    |       | Fall  |       | Rise       |       | Fall  |       |
|               | min     | max   | min   | max   | min        | max   | min   | max   |
| -----         | -----   | ----- | ----- | ----- | -----      | ----- | ----- | ----- |
| middle/p_out  | --      | --    | --    | --    | --         | --    | --    | --    |
| Nets          |         |       |       |       |            |       |       |       |

```

p_in
p_out
n2
n1
n0
```

```
Cells

n3_i
n2_i
n1_i
n0_i
```

```
1
```

---

## Using Ideal Latency

By default, the delay of an ideal network is zero. You can specify ideal latency on pins and ports in an ideal network by using the `set_ideal_latency` command. Ideal latency is accumulated along a path in the same way as a delay value.

Note that ideal latency takes effect only if the object is ideal. If it is not in an ideal network, PrimeTime issues a warning to that effect in the `report_ideal_network` command.

Remove latency from a pin or port by using the `remove_ideal_latency` command.

---

## Using Ideal Transition

The transition time of an ideal network is zero by default. You can, however, specify an ideal transition time value with the `set_ideal_transition` command.

If you set an ideal transition value on an object, the value is propagated from that object along the ideal network either to the network boundary pins or until another ideal transition value is encountered.

Ideal transitions you have annotated on pins or ports take effect only if the object is ideal. If the object is not ideal, PrimeTime issues a warning in the `report_ideal_network` command informing you of this. Use the `remove_ideal_transition` command to remove the transition time you set on a port or pin.

# 8

## SDF Back-Annotation

---

You can back-annotate a design with detailed delay information from a Standard Delay Format (SDF) file for more accurate timing analysis.

SDF back-annotation is described in the following sections:

- [Overview of SDF Back-Annotation](#)
- [Reading SDF Files](#)
- [Reporting Delay Back-Annotation Status](#)
- [Annotating Conditional Delays From SDF](#)
- [Writing an SDF File](#)
- [Removing Annotated Delays and Checks](#)
- [Setting Annotations From the Command Line](#)

---

## Overview of SDF Back-Annotation

For initial static timing analysis, PrimeTime estimates net delays based on a wire load model. Actual delays depend on the physical placement and routing of the cells and nets in the design.

A floor planner or router can provide more detailed and accurate delay information, which you can provide to PrimeTime for a more accurate analysis. This process is called delay back-annotation. Back-annotated information often is provided in an SDF file.

You can read SDF back-annotated delay information in these ways:

- Read the delays and timing checks from an SDF file.
- Annotate delays, timing checks, and transition times from the command line without using the SDF format.

PrimeTime supports SDF v1.0 through 2.1 and a subset of v3.0 features. In general, it supports all SDF constructs except for the following:

- PATHPULSE, GLOBALPATHPULSE
- NETDELAY, CORRELATION
- PATHCONSTRAINT, SUM, DIFF, SKEWCONSTRAINTS

It also supports the following subset of SDF v3.0 constructs:

- RETAIN
- RECREM
- REMOVAL
- CONDELSE

If you do not have an SDF file, you can specify delays in an analyzer script file containing capacitance and resistance parasitics annotation commands. For more information, see [Chapter 9, “Parasitic Back-Annotation.”](#)

---

## Reading SDF Files

The `read_sdf` command reads instance-specific pin-to-pin leaf cell and net timing information from an SDF version 1.0, 1.1, 2.0, 2.1, or 3.0 file, and uses the information to annotate the current design.



Instance names in the design must match instance names in the timing file. For example, if the timing file was created from a design using VHDL naming conventions, the design you specify must use VHDL naming conventions.

---

## Annotating Timing From a Subdesign Timing File

When you specify the `-path` option, the `read_sdf` command annotates the current design with information from a timing file created from a subdesign of the current design. When you specify a subdesign, you cannot use the net delays to the ports of the subdesign to annotate the current design.

---

## Annotating Load Delay

The load delay, also known as extra source gate delay, is the portion of the cell delay caused by the capacitive load of the driven net. Some delay calculators consider the load delay part of the net delay; other delay calculators consider the load delay part of the cell delay. By default, the `read_sdf` command assumes the load delay is included in the cell delay in the timing file being read. If your timing file includes the load delay in the net delay instead of in the cell delay, use the `-load_delay` option with the `read_sdf` command.

---

## Annotating Timing Checks

When the timing file contains the following checks, they are used to annotate the current design.

---

| For check           | SDF construct is           |
|---------------------|----------------------------|
| setup and hold      | SETUP, HOLD, and SETUPHOLD |
| recovery            | RECOVERY                   |
| removal             | REMOVAL                    |
| minimum pulse width | WIDTH                      |
| minimum period      | PERIOD                     |
| maximum skew        | SKEW                       |
| no change           | NOCHANGE                   |

---

---

## Reading the File

The `read_sdf` command reads an SDF file. After reading an SDF file, PrimeTime reports the following:

- Number of errors found while reading the SDF file (for example, pins not found in the design)
- Number of annotated delays and timing checks
- Unsupported SDF constructs found in the SDF file, with the number of occurrences of each SDF construct
- Process, temperature, and voltage values found in the SDF file
- Annotated delays and timing checks of the design (with the `report_annotated_delay` and `report_annotated_check` commands)

For more information, see the `read_sdf` man page.

This command reads from disk the SDF format file `adder.sdf`, which contains load delays included in the cell delays and uses its information to annotate the timing on the current design.

```
pt_shell> read_sdf -load_delay cell adder.sdf
```

These commands read the timing information of instance `u1` of design `MULT16` from the disk file `mult16_u1.sdf`, and annotates the timing on the design `MY_DESIGN`. The load delay is included in the net delays.

```
pt_shell> current_design MY_DESIGN
pt_shell> read_sdf -load_delay net -path u1 mult16_u1.sdf
```

This command reads timing information and annotates the current design with the worst timing when the timing file has different timing conditions for the same pin pair. The load delay is assumed to be included in the cell delay.

```
pt_shell> read_sdf -cond_use_max boo.sdf
```

This command reads minimum and maximum timing information and annotates the current design with delays corresponding to minimum and maximum operating conditions. When reporting minimum delay, PrimeTime uses delays annotated for the minimum condition. When reporting maximum delays, PrimeTime uses delays annotated for the maximum condition.

```
pt_shell> read_sdf -analysis_type bc_wc boo.sdf
```

This command reads minimum and maximum timing information from two separate SDF files and annotates the current design with delays corresponding to minimum and maximum operating conditions. When reporting minimum delays, PrimeTime uses delays annotated for the minimum condition. When reporting maximum delays, PrimeTime uses delays annotated for the maximum condition.

```
pt_shell> read_sdf -analysis_type bc_wc \
 -min_file boo_bc.sdf -max_file boo_wc.sdf
```

---

## Removing Annotated Timing Checks and Delays

To remove timing checks annotated with the `read_sdf` command, use the `reset_design` or `remove_annotated_check` command. To remove delays read and annotated with the `read_sdf` command, use the `reset_design` or `remove_annotated_delay` command.

---

## Managing Large Files

You can compress large files such as SDC, SDF, and script files to a UNIX gzip file when you use the following `write` commands with the `-compress` option:

- `write_sdc`
- `write_sdf`
- `write_sdf_constraints`
- `write_script`
- `write_ilm_script`

### Note:

If you intend to use the `write_sdf_constraints` command as part of your flow, it is recommended that you set the `timing_save_pin_arrival_and_slack` variable to `true` before your first timing update. For more information, see [“Generating Timing Constraints for Place and Route” on page 8-20](#) or the man page.

An example of writing an SDF file to a gzip file is as follows:

```
pt_shell> write_sdf -compress gzip 1.sdf.gz
```

The following commands read gzip compressed files:

- `read_verilog`
- `read_parasitics`
- `read_sdf`

- `read_sdc`

Each command recognizes gzip compressed files automatically. For example, to read a gzip file, enter

```
pt_shell> read_sdf 1.sdf.gz
```

---

## Reporting Delay Back-Annotation Status

Because SDF files are usually large (about 100 MB or larger) and contain many delays, it is a good idea to verify that all nets are back-annotated with delays (and timing checks, where applicable).

---

### Reporting Annotated or Nonannotated Delays

You can check and identify nets that have not been back-annotated with delays in SDF. PrimeTime checks and reports on cell delays and net delays independently. When reporting net delays, PrimeTime considers three types of nets:

- Nets connected to primary input ports
- Nets connected to primary output ports
- Internal nets that are not connected to primary ports

This distinction is important because according to the SDF standard, only internal nets are annotated in SDF.

The `report_annotated_delay` command reports the number of annotated and nonannotated delay arcs. To produce a report of annotated delays, use the `report_annotated_delay` command.

PrimeTime displays a report similar to the following:

| Delay type                   | Total | Annotated | NOT<br>Annotated |
|------------------------------|-------|-----------|------------------|
| cell arcs                    | 16    | 14        | 2                |
| cell arcs (unconnected)      | 5     | 5         | 0                |
| internal net arcs            | 3     | 3         | 0                |
| net arcs from primary inputs | 11    | 11        | 0                |
| net arcs to primary outputs  | 4     | 4         | 0                |
|                              | 39    | 37        | 2                |

For more information, see the `report_annotated_delay` man page.

---

## Reporting Annotated or Nonannotated Timing Checks

You can create a report showing the number of annotated timing checks of all cell timing arcs within your design.

To produce an annotated timing check report, use the `report_annotated_check` command.

PrimeTime displays a report similar to the following:

```

report: annotated_check
Design: my_design

```

|                          | Total | Annotated | NOT<br>Annotated |
|--------------------------|-------|-----------|------------------|
| cell setup arcs          | 2368  | 2368      | 0                |
| cell hold arcs           | 2368  | 2368      | 0                |
| cell recovery arcs       | 676   | 676       | 0                |
| cell removal arcs        | 0     | 0         | 0                |
| cell min pulse width arc | 135   | 135       | 0                |
| cell min period arcs     | 822   | 822       | 0                |
| cell max skew arcs       | 716   | 716       | 0                |
| cell nochange arcs       | 0     | 0         | 0                |
|                          | 7085  | 7085      | 0                |

For more information, see the `report_annotated_check` man page.

---

## Faster Timing Updates in SDF Flows

For each point in a path, PrimeTime calculates slew from the input slew and capacitance and propagates the calculated slew forward from that point in the path. The `timing_use_zero_slew_for_annotated_arcs` variable enables you to use zero slew for arcs annotated with SDF delays, thereby reducing runtime when analyzing designs with all or almost all arcs annotated with SDF.

By default, this variable is set to `auto`. In a design that uses SDF-annotated delays on all arcs or almost all arcs, such as 95% or more, you can forego the higher accuracy of slew calculation in favor of faster runtime with the `auto` setting. In this case, for each arc that is fully annotated by either the `read_sdf` or `set_annotated_delay` command, PrimeTime skips the delay and slew calculation and sets a slew of zero on the load pin of the annotated arc. As a result, timing updates can be completed in significantly less time.

For any arcs that are not annotated, PrimeTime estimates the delay and output slew using the best available input slew. For a block of arcs that are not annotated, PrimeTime propagates the slew throughout the block using the `worst_slew` mode if the `timing_slew_propagation_mode` variable is set to `worst_slew`.

When you use this feature, it is recommended that you disable prelayout slew scaling by setting the `timing_prelayout_scaling` variable to `false`.

---

## Annotating Conditional Delays From SDF

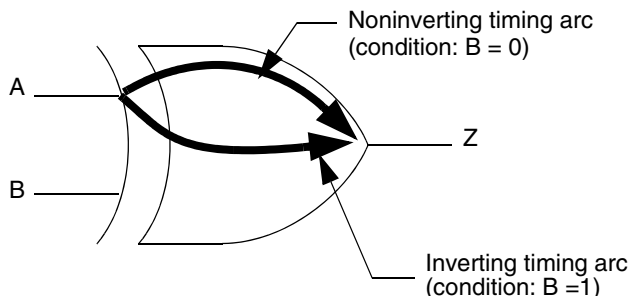
Delays and timing checks specified in an SDF file can be conditional. The SDF condition is usually an expression based on the value of some inputs of the annotated cell. The way PrimeTime annotates these conditional delays depends on whether the Synopsys library specifies conditional delays.

- If the Synopsys library contains conditional arcs, all conditional delays specified in SDF file are annotated. The condition string specified in the Synopsys library with the `sdf_cond` construct must exactly match the condition string in the SDF file.
- If the Synopsys library does not contain conditional arcs (there is no `sdf_cond` construct in the Synopsys library), the maximum or minimum delays of all conditional delays from SDF are annotated. To specify whether to annotate the minimum or maximum delays, use the `-cond_use_max` or `-cond_use_min` option of the `read_sdf` command.

If your library contains state-dependent delays, using a Synopsys library containing conditional arcs enables more accurate annotation from the SDF.

PrimeTime uses the condition specified in the SDF only to annotate the delays and timing checks to the appropriate timing arc specified in the Synopsys library. If you have conditional timing arcs in your SDF, and your library is defined correctly to support conditional arcs, you can use case analysis (or constant propagation) to enable the desired conditional arc values. Consider the example 2-input XOR gate in [Figure 8-1](#).

*Figure 8-1 Example of State-Dependent Timing Arcs*

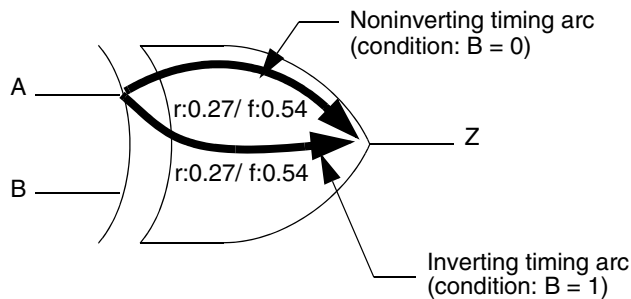


The SDF for the delay from A to Z looks like this:

```
(INSTANCE U1)
(DELAY
 (ABSOLUTE
 (COND B (IOPATH A Z (0.21) (0.54)))
 (COND ~B (IOPATH A Z (0.27) (0.34)))
)
)
```

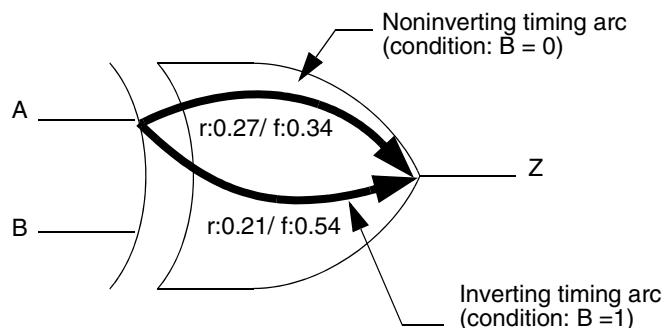
If the Synopsys library contains no conditions, the annotation from SDF uses the worst-case delay for all timing arcs from A to Z. Mapping from the library the timing arc that corresponds to a given condition is not possible. In this case, the annotated delays are as shown in [Figure 8-2](#).

**Figure 8-2** Annotated Delays When the Synopsys Library Contains No Conditions



If the Synopsys library contains conditions, the annotation can identify which timing arc corresponds to the SDF condition. In this case, the annotated delays are as shown in [Figure 8-3](#).

**Figure 8-3** Annotated Delays When the Synopsys Library Contains Conditions



**Note:**

An `IOPATH` statement in the SDF file annotates all arcs between two pins. However, even if an `IOPATH` statement follows a `COND IOPATH` statement, the `COND IOPATH` statement takes precedence over the `IOPATH` statement.

The `IOPATH` delay for the A to Z arc varies depending on whether the B input is a 1 or 0. To select the B = 0 delays, set a case analysis on the B input pin. For example, enter

```
pt_shell> set_case_analysis 0 [get_pins U1/B]
```

Note:

It is possible for a constant to propagate to a pin that selects a conditional arc. This can occur through normal logic constant propagation from a tie-high or tie-low condition, or through a `set_case_analysis` that was set on another pin that propagates a constant to the pin.

---

## Writing an SDF File

You might want to write out the back-annotated delay information to use for gate-level simulation or another purpose. Use the `write_sdf` command to write the delay information in SDF version 1.0, 2.1, or 3.0 format. The default output format is version 2.1. For example, to write an SDF file, enter

```
pt_shell> write_sdf -version 2.1 -input_port_nets mydesign.sdf
```

Note:

If you use a utility other than the `write_sdf` command to write out the SDF file, you should ensure that the annotations are explicitly specified where the SDF version permits.

For more information about the `write_sdf` command, see the man page. For information about specifying the format of the written SDF, see [Appendix A, "Writing Mapped SDF Files."](#)

The SDF written by PrimeTime has the style described in the following sections:

- [SDF Constructs](#)
- [SDF Delay Triplets](#)
- [SDF Conditions and Edge Identifiers](#)
- [Reducing SDF for Clock Mesh/Spine Networks](#)
- [Writing VITAL Compliant SDF Files](#)

---

## SDF Constructs

The SDF written by PrimeTime uses the following SDF constructs:

- `DELAYFILE`, `SDFVERSION`, `DESIGN`, `DATE`, `VENDOR`, `PROGRAM`, `VERSION`, `DIVIDER`, `VOLTAGE`, `PROCESS`, `TEMPERATURE`, `TIMESCALE`



- CELL, CELLTYPE, INSTANCE
- ABSOLUTE, COND, CONDELSE, COSETUP, DELAY, HOLD, INTERCONNECT, IOPATH, NOCHANGE, PERIOD, RECOVERY, RECREM, RETAIN, SETUP, SETUPHOLD, SKEW, TIMINGCHECK, WIDTH

**Note:**

The following constructs are supported in SDF version 3.0 only: CONDELSE, RETAIN, RECREM, REMOVAL.

- Posedge and negedge identifiers

The SDF written by the `write_sdf` command does not use the following SDF constructs: INCREMENT, CORRELATION, PATHPULSE, GLOBALPATHPULSE, PORT, DEVICE, SUM, DIFF, SKEWCONSTRAINT, PATHCONSTRAINT. However, the `write_sdf_constraints` command supports the `PATHCONSTRAINT` construct.

---

## SDF Delay Triplets

The SDF delay triplet values depend on whether your design uses a single operating condition or minimum and maximum operating conditions. For a single operating condition, the SDF delay triplet has three identical values, for example: (1.0:1.0:1.0).

For minimum and maximum operating conditions, the SDF triplet contains only two delays for the minimum operating condition and the maximum operating condition, respectively: (1.0::2.0). The typical delay of the SDF triplet is not used. The SDF delays written by PrimeTime specify the following transitions: 0 to 1, 1 to 0, 0 to Z, Z to 1, 1 to Z, and Z to 0.

---

## SDF Conditions and Edge Identifiers

PrimeTime takes advantage of the edge identifiers as well as conditions if edge identifiers are specified in the library with `sdf_cond`. PrimeTime uses edge identifiers for timing checks and cell delays.

PrimeTime writes an edge identifier (`POSEDGE` or `NEGEDGE`) for a combinational cell delay arc when the positive edge delay differs from the negative edge delay, and the input net transition differs between rise and fall on the input pin of the delay arc. As a result, two timing arc `IOPATH` delays can be generated for a given timing arc. For example,

```
(CELL
 (CELLTYPE "XOR")
 (INSTANCE U1)
 (DELAY
 (ABSOLUTE
 (IOPATH (posedge A) Z (0.936:0.936:0.936)
(1.125:1.125:1.125))
 (IOPATH (negedge A) Z (1.936:1.936:1.936)
(2.125:2.125:2.125))
```

```

 (IOPATH (posedge B) Z (0.936:0.936:0.936)
(1.125:1.125:1.125))
 (IOPATH (negedge B) Z (1.936:1.936:1.936)
(2.125:2.125:2.125))
)
)
)

```

This is common for library cells such as multiplexers and exclusive-OR cells. Other SDF writers might only generate one set of delay triplets for the positive and negative edges. PrimeTime writes both for the highest accuracy. However, some logic simulators do not support edge identifiers on combinational and sequential timing arcs and expect to see only one timing arc. To write an SDF that is compatible with these simulators, use the `-no_edge` option with the `write_sdf` command. For example,

```
pt_shell> write_sdf -no_edge mydesign.sdf
```

With the `-no_edge` option, PrimeTime generates only one timing arc, with the worst-case delay triplets for the positive and negative transitions. Note that the simulation timing delays might be pessimistic as a result of using the `-no_edge` option.

---

## Reducing SDF for Clock Mesh/Spine Networks

A design with a large clock mesh or spine network can produce an unreasonably large SDF file because of the extremely large number of nets in the clock network. In these cases, you can choose to have PrimeTime reduce the SDF file by combining SDF values that differ by a negligible amount, and using the SDF 3.0 PORT construct to represent the combined nets.

Note:

For related information, see [“Fast Multidrive Delay Analysis” on page 5-40](#) and [“Parallel Driver Reduction” on page 5-41](#).

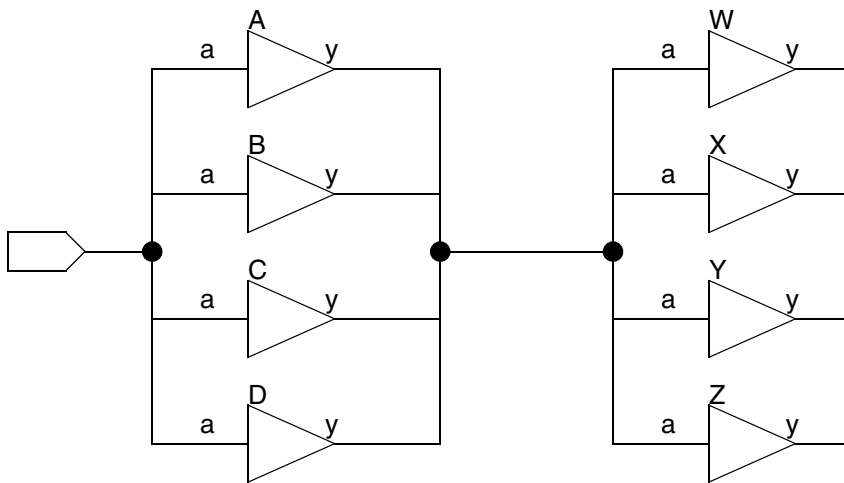
The SDF reduction features operate under the control of four variables:

- `sdf_enable_port_construct` – Boolean variable, when set to `true`, enables the PORT construct to be used for writing SDF. The default setting is `false`.
- `sdf_enable_port_construct_threshold` – Floating-point value that specifies the absolute delay difference, in picoseconds, below which the PORT construct is used. The default setting is 1 ps.
- `sdf_align_multi_drive_cell_arcs` – Boolean variable, when set to `true`, causes PrimeTime to unify the small differences in driver cell outputs to networks that constitute a mesh or spine, which can cause simulation failure. The default setting is `false`.
- `sdf_align_multi_drive_cell_arcs_threshold` – Floating-point value that specifies the absolute delay difference, in picoseconds, below which multidrive arcs are aligned. The default setting is 1 ps.

## PORT Construct

The `sdf_enable_port_construct` variable determines whether the PORT statement is used to replace multiple INTERCONNECT statements. The PORT statement is used in all parallel nets (within specified threshold) in the design except those parallel nets which are driven by three-state buffers. If the PORT statement is used, the `sdf_enable_port_construct_threshold` variable determines the maximum allowable absolute difference in delay arc values for interconnections to be combined. For example, consider the clock network shown in [Figure 8-4](#).

Figure 8-4 Parallel Buffers Driving Parallel Buffers



If using the PORT construct is disabled (the default), the `write_sdf` command writes out this clock network using SDF syntax similar to the following:

```
(INSTANCE top)
(DELAY
 (ABSOLUTE
 (INTERCONNECT A/y W/a (0.01 ::0.03))
 (INTERCONNECT A/y X/a (0.02 ::0.04))
 (INTERCONNECT A/y Y/a (0.01 ::0.04))
 (INTERCONNECT A/y Z/a (0.02 ::0.03))
 (INTERCONNECT B/y W/a (0.01 ::0.03))
 (INTERCONNECT B/y X/a (0.03 ::0.05))
 ...
 (INTERCONNECT D/y Y/a (0.02 ::0.05))
 (INTERCONNECT D/y Z/a (0.01 ::0.03))
)
)
```

There are 16 interconnection combinations listed.

If the PORT construct is enabled and if the variation in delay values is within the specified threshold, the `write_sdf` command reduces the SDF as follows:

```
(INSTANCE top)
(Delay
 (ABSOLUTE
 (PORT W/a (0.02 ::0.04))
 (PORT X/a (0.03 ::0.05))
 (PORT Y/a (0.02 ::0.06))
 (PORT Z/a (0.03 ::0.07))
)
)
```

Four PORT statements replace 16 INTERCONNECT statements.

## Normalizing Multidriven Arcs for Simulation

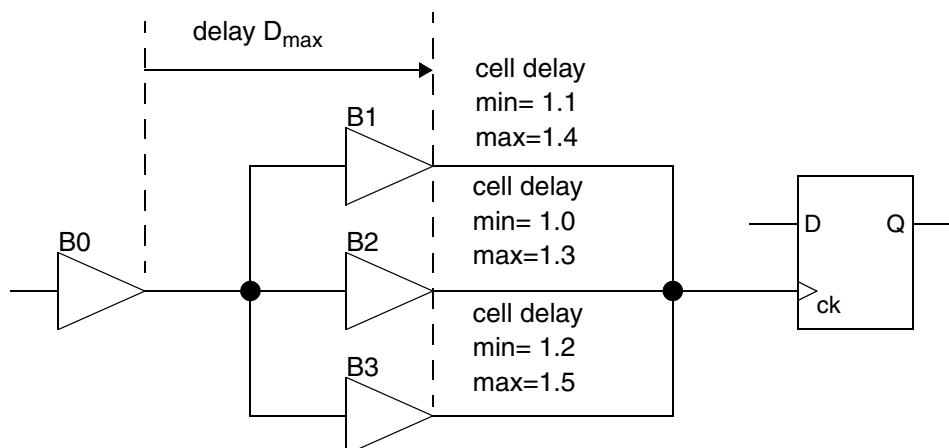
When the `sdf_align_multi_drive_cell_arcs` and `sdf_enable_port_construct` variables are enabled, PrimeTime aligns similar delay values of cell timing arcs where multiple drivers drive a common net.

### Note:

If the common net is driven by three-state buffers, PrimeTime does not align cell delays.

The `sdf_align_multi_drive_cell_arcs_threshold` variable then determines the maximum allowable absolute difference in delay arc values for normalizing to occur. Normalizing means using a single worst-case delay arc value to represent multiple drivers. This can prevent errors from occurring when the SDF file is used for circuit simulation. For example, consider the cell delays in the parallel driver network shown in [Figure 8-5](#).

*Figure 8-5 Cell Delays in a Parallel Driver Network*



PrimeTime aligns the cell arcs if the `sdf_align_multi_drive_cell_arcs` variable is set to `true`, the delay are values are within the `sdf_align_multi_drive_cell_arcs_threshold` variable setting, the cells in the clock network are combinational (not sequential), and each cell has one input and one output.

The normalizing process adjusts both the net delays and cell delays in the clock network as needed to ensure that the complete path delays are the same. Net delays are adjusted only if the `sdf_enable_port_construct` variable is set to `true` and the delays are within the threshold variable setting, in which case the PORT statement is used to represent the net arcs connected to the load pin “ck” in the example. In the case of the nets connecting B0 to cells B1 through B3, no changes are made to the delay values because each of the three cells has only one fanin net.

In the case of the cell delays, the worst delay from B0 to the output pin of cells B1 through B3 is calculated to be Dmax. The cell arcs of B1 through B3 are adjusted to make the delay of each path from B0 to the output pins of the cells to be equal to Dmax. In this example, the cells are given the same delays because the net arcs from B0 to the cells B1 through B3 all have the same delays. Taking the worst delay means using the minimum of min delays and maximum of max delays, the most pessimistic values. In the case of cells with multiple arcs, the worst cell arc is used to represent the cell delay.

The `sdf_align_multi_drive_cell_arcs` variable should be set to `true` only to generate SDF for simulation. Avoid reading the generated SDF back into PrimeTime for timing analysis, as the data is pessimistic.

If multidrive normalizing is disabled, the `write_sdf` command writes the following description of the example network.

#### Net delays:

```
(INTERCONNECT B1/z FF1/ck (0.01 ::0.04))
(INTERCONNECT B2/z FF1/ck (0.03 ::0.05))
(INTERCONNECT B3/z FF1/ck (0.02 ::0.04))
```

#### Cell delays:

```
(CELLTYPE "buf")
(INSTANCE B1)
(DELAY
 (ABSOLUTE
 (DEVICE (1::8) (4::7))
)
)
(CELLTYPE "buf")
(INSTANCE B2)
(DELAY
 (ABSOLUTE
 (DEVICE (2::9) (3::11))
)
)
```

```

)
)
 (CELLTYPE "buf")
 (INSTANCE B3)
 (DELAY
 (ABSOLUTE
 (DEVICE (1::5) (5::8))
)
)
)

```

If multidrive normalizing is enabled, and if the delay differences are under the specified threshold, the `write_sdf` command writes the following description of the example network.

#### Net delays:

```
(PORT FF1/ck (0.03 ::0.05))
```

#### Cell delays:

```

(CELLTYPE "buf")
(INSTANCE B1)
(DELAY
 (ABSOLUTE
 (DEVICE (2::9) (5::11))
)
)
(CELLTYPE "buf")
(INSTANCE B2)
(DELAY
 (ABSOLUTE
 (DEVICE (2::9) (5::11))
)
)
(CELLTYPE "buf")
(INSTANCE B3)
(DELAY
 (ABSOLUTE
 (DEVICE (2::9) (5::11))
)
)

```

---

## Writing VITAL Compliant SDF Files

If you want to write out SDF that is VITAL compliant, use the following command:

```
pt_shell> write_sdf -no_edge_merging -exclude {"no_condelse"} \
 file.sdf
```

If the simulator you plan to use cannot handle negative delays, you can use the `-no_negative_values` option with the `timing_checks`, `cell_delays`, or `net_delays` values. You should use the `-no_negative_values timing_checks` option when you want

to zero out all negative timing check values, such as setup, hold, recovery or removal SDF statements. Use the `-no_negative_values cell_delays` option when you want to zero out all negative cell delay values, such as IOPATH and port SDF statements. Use the `-no_negative_values net_delays` option when you want to zero out all negative net delays, such as interconnect SDF statements. For example, if you want to zero out timing checks and net delays, use the following syntax:

```
pt_shell> write_sdf -no_edge_merging -no_negative_values \
 {timing_checks net_delays} -exclude {"no_condelse"} file.sdf
```

---

## Removing Annotated Delays and Checks

You can remove annotated delays and timing checks using the `remove_annotated_delay` and `remove_annotated_check` commands. For more information, see the following sections:

- [Removing Annotated Delays](#)
- [Removing Annotated Checks](#)

---

### Removing Annotated Delays

The `remove_annotated_delay` command removes annotated cell and net delays from the current design. Delays are annotated either from an SDF file by using the `read_sdf` or `set_annotated_delay` commands.

To remove all annotated net and cell delays from the current design, enter

```
pt_shell> remove_annotated_delay -all
1
```

To remove annotated delays from some cells, enter

```
pt_shell> remove_annotated_delay [get_cells u1*]
1
```

The following command removes annotated delays between pins. When there are no annotated delays between the specified pins, a warning message appears.

```
pt_shell> remove_annotated_delay -from ffb/Q
1
pt_shell> remove_annotated_delay -from ffb/Q -to u1/A
Warning: No annotated delays from 'ffb/Q' to 'u1/A'
0
```

For more information, see the `remove_annotated_delay` man page.

---

## Removing Annotated Checks

The `remove_annotated_check` command removes annotated timing checks from cells in the current design. Timing checks are annotated either from an SDF file by using the `read_sdf` command, or the `set_annotated_check` command. Timing check types include setup, hold, recovery, removal, and no-change.

To remove all annotated cell timing checks from the current design, enter

```
pt_shell> remove_annotated_check -all
```

To remove only setup checks from all cells in the current design, enter

```
pt_shell> remove_annotated_check -setup [get_cells *]
1
```

The following command removes checks between pins. The error condition is shown when the pins are not on the same cell, and when there are no annotated checks, a warning message appears.

```
pt_shell> remove_annotated_check -from ffb/CP -to ffa/D
Error: Cannot remove annotated check from 'ffb/CP' to 'ffa/
D':
pins are on different cells (PTE-032)
0
pt_shell> remove_annotated_check -from ffa/CP -to ffa/D \
-setup -hold
1
pt_shell> remove_annotated_check -from ffa/CP -to ffa/D
Warning: No annotated timing checks were removed. (PTE-031)
0
```

For more information, see the `remove_annotated_check` man page.

---

## Setting Annotations From the Command Line

You can annotate delays, timing checks, and transition times from the command line to make a limited number of changes for debugging. To manually set annotations, use these commands:

- `set_annotated_delay`
- `set_annotated_check`
- `set_annotated_transition`



---

## Annotating Delays

The `set_annotated_delay` command sets cell or net delays from the command line. To list annotated delay values, use the `report_annotated_delay` command. To set a cell delay, you specify the delay from a cell input to an output of the same cell. To set a net delay, you specify the delay from a cell output to a cell input.

To remove the annotated cell or net delay values from a design, use the `remove_annotated_delay` or `reset_design` command.

This example annotates a cell delay of 20 units between input pin A of cell instance U1/U2/U3 and output pin Z of the same cell instance. The delay value of 20 includes the load delay.

```
pt_shell> set_annotated_delay -cell -load_delay cell 20 \
 -from U1/U2/U3/A -to U1/U2/U3/Z
```

This example annotates a rise net delay of 1.4 units between output pin U1/Z and input pin U2/A. The delay value for this net does not include load delay.

```
pt_shell> set_annotated_delay -net -rise 1.4 -load_delay \
 cell -from U1/Z -to U2/A
```

This example annotates a rise net delay of 12.3 units between the same output pins. In this case the net delay value does include load delay.

```
pt_shell> set_annotated_delay -net -rise 12.3 -load_delay \
 net -from U1/Z -to U2/A
```

This example annotates a fall cell delay of 21.2 units on the enable arc of the three-state cell instance U8.

```
pt_shell> set_annotated_delay -cell -fall -of_objects \
 [get_timing_arcs -from U8/EN -to U8/Z \
 -filter sense==enable_low] 21.2
```

---

## Annotating Timing Checks

The `set_annotated_check` command sets the setup, hold, recovery, removal, or no-change timing check value between two pins.

To list annotated timing check values, use the `report_annotated_check` command. To remove annotated timing check values from a design, use the `remove_annotated_check` or `reset_design` command. To see the effect of `set_annotated_check` for a specific instance, use the `report_timing` command.

This example annotates a setup time of 2.1 units between clock pin CP of cell instance u1/ff12 and data pin D of the same cell instance.

```
pt_shell> set_annotated_check -setup 2.1 -from u1/ff12/CP \
 -to u1/ff12/D
```

## Annotating Transition Times

The `set_annotated_transition` command sets the transition time on any pin of a design. Transition time (also known as slew) is the amount of time it takes for a signal to change from low to high or from high to low.

This example annotates a rising transition time of 0.5 units and a falling transition time of 0.7 units on input pin A of cell instance U1/U2/U3.

```
pt_shell> set_annotated_transition -rise 0.5 [get_pins U1/U2/U3/A]
pt_shell> set_annotated_transition -fall 0.7 [get_pins U1/U2/U3/A]
```

---

## Generating Timing Constraints for Place and Route

Use the `write_sdf_constraints` command to write path timing constraints in SDF versions 1.0 and 2.1 formats to a constraints file. Use the constraint file to constrain layout tools to meet critical timing goals.

To write constraints for the current design, arrival totals and slacks must be available throughout the design, not just at endpoints. The optimal flow (in terms of CPU usage) to use this command within PrimeTime is as follows:

1. Set the `timing_save_pin_arrival_and_slack` variable to `true` to store slacks at all pins in the design.
2. Update timing by using the `update_timing` command.

Note:

If you intend to use `write_sdf_constraints` command as part of your flow, it is recommended that you set the `timing_save_pin_arrival_and_slack` variable to `true` before your first timing update.

3. Use the `write_sdf_constraints` to write constraints for the current design.

When you use the `write_sdf_constraints` command, keep the following limitations in mind:

- PrimeTime does not generate capacitance constraint files.
- PrimeTime does not support some Design Compiler `write_constraints` command options, including `-cover_nets` and `-load_delay net | cell`.

For more information, see the `write_sdf_constraints` man page.

The following example shows a timing constraint file in SDF v2.1 format.

```
(DELAYFILE
(SDFVERSION "OVI 2.1")
(DESIGN "counter")
(DATE "Thur Nov 30 16:09:53 2000")
(VENDOR "lsi_10k")
(PROGRAM "Synopsys PrimeTime")
(VERSION "2000.11")
(DIVIDER /)
(VOLTAGE 5.00:5.00:5.00)
(PROCESS "NOMINAL")
(TEMPERATURE 25.00:25.00:25.00)
(TIMESCALE 1ns)
(CELL
 (CELLTYPE "counter")
 (INSTANCE)
 (TIMINGCHECK
 (PATHCONSTRAINT ffb/CP ffb/QN w/B w/Z q/A q/Z j/C j/Z
 ffd/D
 (9.100:9.100:9.100) (9.100:9.100:9.100))
 (PATHCONSTRAINT ffb/CP ffb/QN v/A v/Z r/A r/Z g/C g/Z
 ffc/D
 (9.100:9.100:9.100) (9.100:9.100:9.100))
 (PATHCONSTRAINT ffa/CP ffa/QN u/A u/Z s/A s/Z d/C d/Z
 ffb/D
 (9.100:9.100:9.100) (9.100:9.100:9.100))
 (PATHCONSTRAINT ffa/CP ffa/QN t/B t/Z a/C a/Z ffa/D
 (9.150:9.150:9.150) (9.150:9.150:9.150))
)
)
)
```

The following example shows the syntax to write a timing constraint file in SDF v2.1 format for the most critical path in each path group in the current design:

```
pt_shell> write_sdf_constraints cstr.sdf
```

The following example shows the syntax to write a timing constraint file in SDF v1.0 format for the 10 most critical paths in each path group in the current design (named counter):

```
pt_shell> write_sdf_constraints -max_paths 10 \
 -version 1.0 counter_cstr.sdf
```

---

## Providing Constraint Coverage for the Entire Design

To constrain a design fully for placement, every net or every net arc (connection between two cells) generally requires a constraint. You can ensure that a design is fully constrained by forcing the `write_sdf_constraints` command to generate constraints for all paths in the design. Generation of the resulting constraint file requires a large amount of CPU time and disk space.

Some place and route tools require only the worst path through any point (other data is not used). In these cases, generating the worst path through every driver-load pin pair in the design is sufficient to provide full constraint coverage. In that case, the total number of paths for the design is less than or equal to the total number of leaf cell input pins plus the number of primary outputs (often, points are covered by multiple paths).

The `write_sdf_constraints` command's `-cover_design` option generates just enough unique paths to provide constraint coverage for the entire design. The overall runtime with the `-cover_design` option is greater, but memory and disk space requirements are significantly less.

### Note:

The `-cover_design` option ensures that a constraint is placed on every driver-load pin pair in the design, but does not fully constrain the design. The coverage is minimal; it contains less information than a constraint file with all paths enumerated. Theoretically, a place and route tool can meet all the given constraints and still have timing violations.

The `-cover_design` option is most useful for large designs in which generating data for all paths is not reasonable, or when the targeted place and route tool uses only the worst path through any point. A point can be a net or an edge, where an edge is any connection between two leaf-level cells.

For large designs in which generating data for all paths is not reasonable, using the `-cover_design` option might be the only way to ensure that every net is constrained at least once. When the targeted place and route tool uses only the worst path through any point, constraints other than the worst are ignored by the place and route tool.

To write a timing constraint file in SDF v2.1 format with just enough paths to cover the worst path through every driver-load pin pair, enter

```
pt_shell> write_sdf_constraints -cover_design \
 counter_cstr.sdf
```

# 9

## Parasitic Back-Annotation

---

You can back-annotate a design with detailed parasitic information (resistance and capacitance) for more accurate timing analysis.

Parasitic back-annotation is described in the following sections:

- [Parasitic Data](#)
- [Lumped Parasitics](#)
- [Reduced and Detailed Parasitics](#)
- [Reading Parasitics Files](#)
- [Incomplete Annotated Parasitics](#)
- [Reporting Annotated Parasitics](#)
- [Removing Annotated Parasitics](#)

---

## Parasitic Data

PrimeTime allows parasitic back-annotation of detailed circuit delay information in the form of lumped capacitance, lumped resistance, reduced pi model, or detailed RC network. The reduced pi model and detailed RC network are more accurate than lumped capacitance and lumped resistance, but using them requires setting environment variables and uses more CPU time and memory.

### Note:

To reduce CPU time usage to a minimum, use an SDF file rather than parasitic data to back-annotate the design. In that case, PrimeTime does not have to calculate the delays. For more information, see [Chapter 8, “SDF Back-Annotation.”](#)

In case of conflict between different types of parasitic data, PrimeTime uses the data on each net in the following order:

1. Lumped resistance and capacitance values annotated with the `set_resistance` and `set_load` commands, if any.
2. Detailed parasitics annotated with the `read_parasitics` command, if any.
3. Wire load models obtained from the technology library or specified with the `set_wire_load_model` command.

Setting lumped resistance or capacitance with the `set_resistance` or `set_load` command temporarily overrides the wire load model or detailed parasitics for a net. Removing lumped resistance or capacitance from a net with the `remove_resistance` or `remove_capacitance` command causes the net to revert back to its previous form of parasitic data.

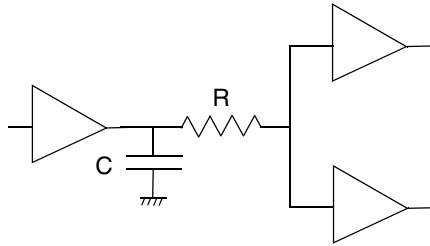
You can set lumped resistance and capacitance separately. For example, you can read in detailed parasitics for a net, and then override just the capacitance for that net with the `set_load` command. In that case, PrimeTime still uses the detailed parasitic information to calculate the net resistance.

---

## Lumped Parasitics

You can annotate resistance and capacitance (RC) on nets, as shown in [Figure 9-1](#). Even if you annotated all the delays of the design with SDF, you might want to annotate parasitics to perform certain design rule checks, such as maximum transition or maximum capacitance.

Figure 9-1 Lumped RC



---

### Setting Net Capacitance

The `set_load` command sets the net capacitance values on ports and nets within the design. If the current design is hierarchical, you must link it with the `link` command.

By default, the total capacitance on a net is the sum of all of pin, port, and wire capacitance values associated with the net. A capacitance value you specify overrides the internally estimated net capacitance.

Use the `set_load` command for nets at lower levels of the design hierarchy. Specify these nets as BLOCK1/BLOCK2/NET\_NAME.

If you use the `-wire_load` option, the capacitance value is set as a wire capacitance on the specified port and the value is counted as part of the total wire capacitance (not as part of the pin or port capacitance).

To view capacitance values on ports, use `report_port`. To view capacitance values on nets, use the `report_net` command. For more information, see the `set_load` man page.

---

### Setting Net Resistance

The `set_resistance` command sets net resistance values in the design. The specified resistance value overrides the internally estimated net resistance value.

You can also use the `set_resistance` command for nets at lower levels of the design hierarchy. You can specify these nets as BLOCK1/BLOCK2/NET\_NAME.

To view resistance values, use the `report_net` command. To remove resistance values annotated on specified nets, use the `remove_resistance` command. To remove resistance annotated on the entire design, use the `reset_design` command.

---

## Reduced and Detailed Parasitics

PrimeTime provides features that enable you to annotate reduced parasitics and detailed parasitics.

---

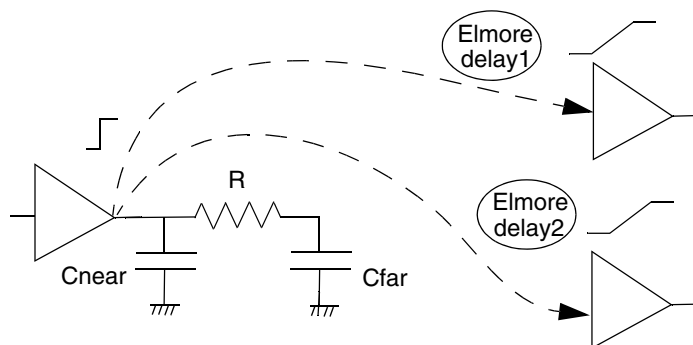
### Annotating Reduced Parasitics

Reduced resistance and capacitance represents the RC within a design from a driver standpoint: two capacitors and one resistor, as shown in [Figure 9-2](#). The net delays are typically annotated with an Elmore delay, which is the most commonly used representation. For this model the following rules apply:

- RC is represented in terms of a pi model (two C and one R).
- Net delay from the driver to the net fanout provided is an Elmore delay.
- The RC pi model is used to compute the slew at the driver pin.
- Slew degradation (transition time) from the driver pin to each net load pin is computed if the Elmore delay is significant (more than 20 percent of driver transition time). The slew degradation is based on a published paper by E. G. Friedman and J. H. Mulligan, Jr., titled "Ramp Input Response of RC Tree Network," in *Analog Integrated Circuits and Signal Processing*, Volume 14, No. 1/2, pp. 53-58, September 1997.

The model in [Figure 9-2](#) calculates the cell delay and the driver output slew more accurately than the lumped RC model. Reduced RC is annotated with RSPF (Reduced Standard Parasitic Format) or SPEF (Standard Parasitic Exchange Format).

*Figure 9-2 Reduced RC*



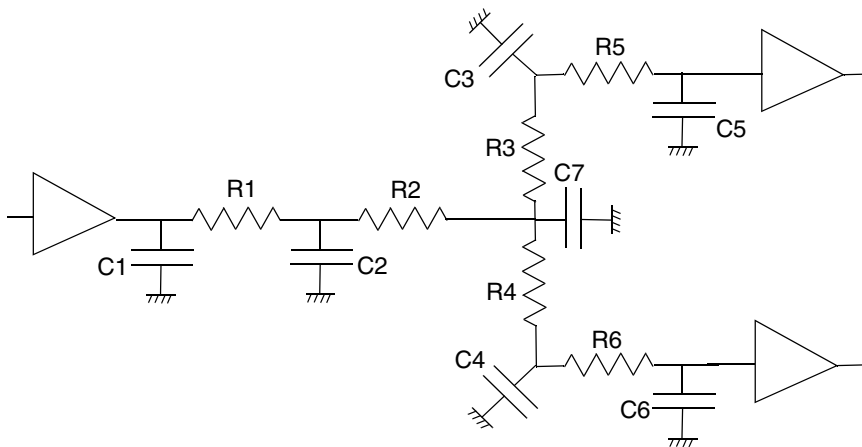


## Annotating Detailed Parasitics

You can annotate detailed parasitics into PrimeTime and annotate each physical segment of the routed netlist in the form of resistance and capacitance (see [Figure 9-3](#)). Annotating detailed parasitics is very accurate but more time-consuming than annotating lumped parasitics. Because of the potential complexity of the RC network, PrimeTime takes longer to calculate the pin-to-pin delays in the netlist.

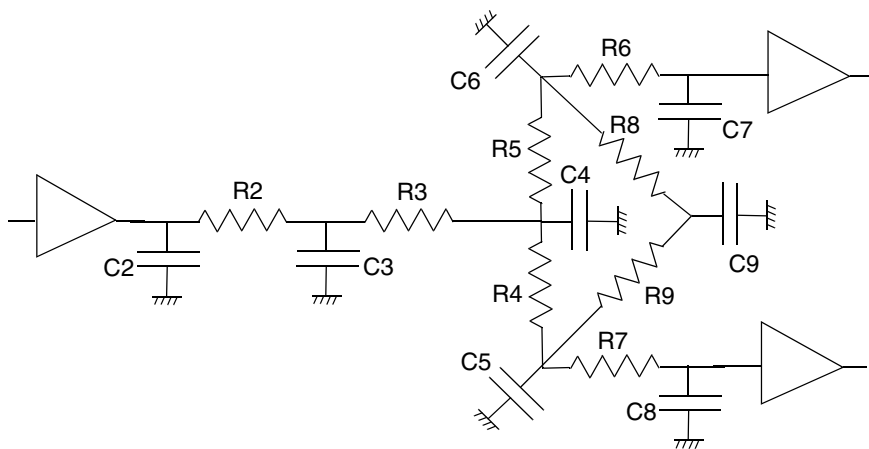
This RC network is used to compute effective capacitance ( $C_{\text{effective}}$ ), slew, and delays at each subnode of the net. PrimeTime can read detailed RC in SPEF format.

*Figure 9-3 Detailed RC*



Use this model for netlists that have critical timing delays, such as clock trees. This model can produce more accurate results, especially in deep submicron designs where net delays are more significant compared to cell delays. The detailed RC network supports meshes, as shown in [Figure 9-4](#).

*Figure 9-4 Meshed RC*



---

## Supported File Formats for Parasitic Annotation

PrimeTime supports the open Verilog International (OVI) SPEF format for parasitic annotation.

The following limitations apply to RSPF and DSPF constructs:

- SPICE inductors (Lxxx) and lines (Txxx) are allowed in the DSPF file, but they are ignored.
- Physical coordinates of pins and instances are ignored.
- The `BUSBIT` construct is not supported.
- Instances listed in the RSPF and DSPF files in the SPICE section are ignored. They are not checked to determine whether they match the current design loaded in PrimeTime.
- Resistors cannot be connected to ground. PrimeTime ignores such resistors and displays a warning about them.
- Capacitors must be connected to ground. PrimeTime ignores node-to-node coupling capacitors and displays a warning about them.

The following limitations apply to SPEF constructs:

- Inductors are ignored.
- Poles and residue descriptions on reduced nets are not supported.
- Resistors cannot be connected to ground. PrimeTime ignores resistors connected to ground and displays a warning.
- Cross-coupling capacitors are split to ground unless you are using PrimeTime SI and crosstalk analysis is enabled.

---

## Characterization Trip Points

The characterization trip points of a design (waveform measurement thresholds) affect the calculation of delays and transition times by PrimeTime. PrimeTime establishes the trip points as follows:

1. If pin-level thresholds are defined, the application tools use those. Pin-level thresholds override library-level thresholds with the same name.
2. If pin-level thresholds are not defined, but library-level thresholds are, the application tools use the library-level thresholds.

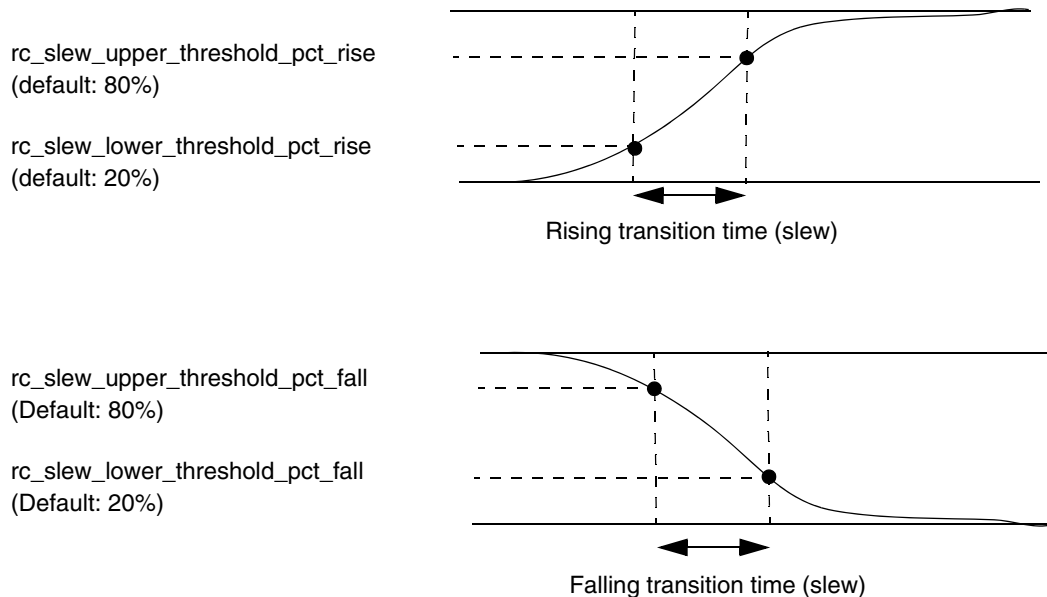
3. If neither pin-level nor library-level thresholds are defined, and the PrimeTime environment variables are not set, the application tools use the thresholds defined in the main library (the first library in the link path).
4. If neither pin-level nor library-level thresholds are defined, the thresholds are determined by PrimeTime environment variables.
5. If none of the above are defined, the application tools use default trip points at 20 percent and 80 percent of the rail voltage for transition time calculations and 50 percent of the rail voltage for delay calculations.
6. If the trip points defined by any of these methods are not valid (for instance, 0 percent and 100 percent), the trip points is set to 5 percent and 95 percent of the rail voltage for transition time calculations and 50 percent of the rail voltage for delay calculations.

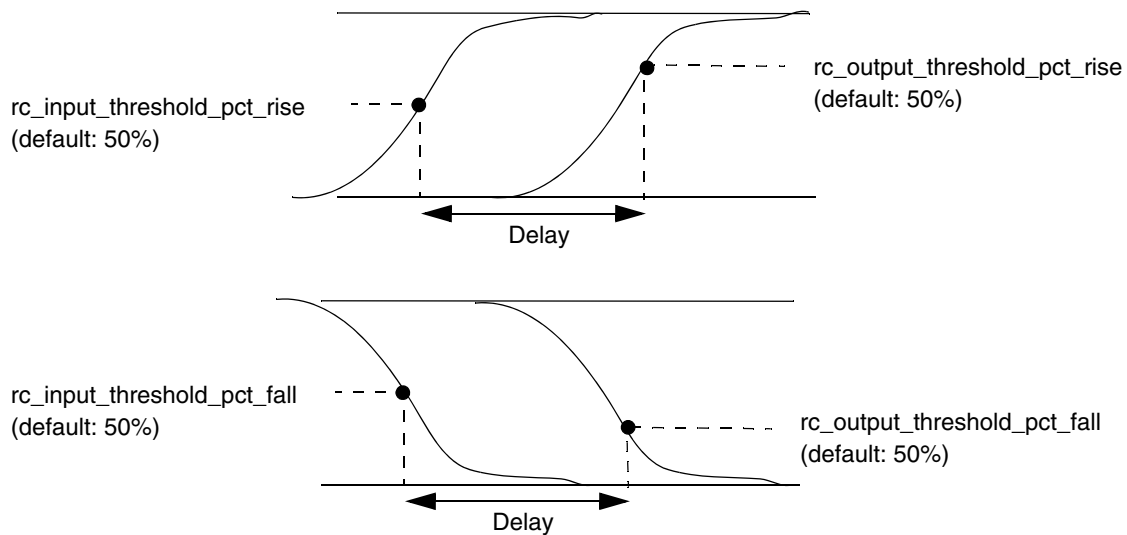
**Note:**

In earlier releases of PrimeTime, the trip points were obtained in a different manner. To revert to this earlier behavior, set the `lib_thresholds_per_lib` variable to `false`. For more information, see the man page for the variable.

Figure 9-5 and Figure 9-6 show some signal waveforms, the trip points used to calculate transition times (slews) and delays, and the variables that can be used to define the trip points. The default trip point values define slew as the time that a signal takes to change from 20 to 80 percent or from 80 to 20 percent of the rail voltage, and they define cell delay as the amount of time from the input signal reaching 50 percent of the rail voltage to the output signal reaching 50 percent of the rail voltage.

**Figure 9-5 Slew Transition Points**



*Figure 9-6 Input/Output Transition Points*

The following table lists the variables that can be used to specify the trip points in the absence of trip point definitions in the technology libraries.

| Variable                         | Default |
|----------------------------------|---------|
| rc_slew_lower_threshold_pct_rise | 20      |
| rc_slew_lower_threshold_pct_fall | 20      |
| rc_slew_upper_threshold_pct_rise | 80      |
| rc_slew_upper_threshold_pct_fall | 80      |
| rc_slew_derate_from_library      | 1       |
| rc_input_threshold_pct_fall      | 50      |
| rc_input_threshold_pct_rise      | 50      |
| rc_output_threshold_pct_fall     | 50      |
| rc_output_threshold_pct_rise     | 50      |

To get the current value of a variable, use the `printvar` command.

To check the threshold levels defined in a library, use the `report_lib` command. To check the threshold settings used for a particular delay calculation arc, use the command `report_delay_calculation -thresholds`.

In the absence of trip point definitions in the libraries used in the design, the following commands specify a set of threshold levels that PrimeTime uses to calculate delays and slews:

```
pt_shell> set rc_slew_derate_from_library 0.4
pt_shell> set rc_slew_lower_threshold_pct_fall 30
pt_shell> set rc_slew_lower_threshold_pct_rise 30
pt_shell> set rc_slew_upper_threshold_pct_fall 70
pt_shell> set rc_slew_upper_threshold_pct_rise 70
pt_shell> set rc_input_threshold_pct_rise 50
pt_shell> set rc_input_threshold_pct_fall 50
```

The four “slew threshold” variables specify that slews were characterized by measuring the transition times from 30 to 70 percent and from 70 to 30 percent of the rail voltage. The “slew derate” variable, which is set to 0.4, specifies that the transition times were extrapolated to the rail voltages (0 to 100 percent of the supply voltage); the range of 30 to 70 percent is a span of 40 percent of the supply voltage. The two “input threshold” and two “output threshold” variables specify that delays were calculated from trip points at 50 percent of the rail voltage.

---

## Reading Parasitics Files

The `read_parasitics` command reads a parasitic data file in SPEF, RSPF (version IEEE 1481-1999), or SBPF (Synopsys Binary Parasitic Format) format, and it annotates the current design with the parasitic information. An SPEF or RSPF file is an ASCII file that can be compressed with gzip. Specifying the format in the command is optional because the reader can automatically determine the file type.

Net and instance pin names in the design must match the names in the parasitics file. For example, if you create the parasitics file from a design using VHDL naming conventions, the design name must use VHDL naming conventions.

When reading parasitics files, by default, PrimeTime assumes that capacitance values specified in the SPEF files do not include the pin capacitance. The pin capacitance values used by PrimeTime are the values specified in the Synopsys design libraries; any pin capacitance values specified in SPEF are ignored. You must ensure that the coupling capacitance in the SPEF file are symmetric. To verify that the coupling capacitance contains only symmetric coupling, read in the design and then use both the `-syntax_only` and `-keep_capacitance_coupling` options of the `read_parasitics` command. PrimeTime checks for asymmetric coupling and issues warning messages when this type of issue is identified. Ensure the SPEF files contain valid coupling capacitance before proceeding.

The reduced and detailed RC networks specified in SPEF files are used to compute effective capacitance dynamically during delay calculation. The capacitance value reported by most report commands, such as the `report_timing` and `report_net` command, is the lumped capacitance, also known as Ctotal. Ctotal is the sum of all capacitance values of a net as specified in the SPEF, to which pin capacitance is also added. To optimize performance for the `read_parasitics` command, index a gzip SPEF design file with the `preload_parasitics` command. For more information, see the `read_parasitics` man page.

Parasitic data files are often very large and time-consuming for PrimeTime to read. To minimize the read time, make sure that the data file is on a local disk that can be accessed directly, not across a network. To avoid using disk swap space, having enough memory is also helpful. Compressing an SPEF file using gzip can improve overall processing time because the file is so much smaller.

The most efficient format is SBPF; consider using it if your tools support it. For example, to read parasitics for the current design from a file called `sub_design1.rspf` using the RSPF file format, enter

```
pt_shell> read_parasitics -format rspf sub_design1.rspf
```

To optimize performance, it is recommended that you create a single parasitics file in the SBPF format. You can use the following syntax:

```
pt_shell> write_parastics -format SBPF file_name
```

You can read multiple parasitics files. For example, you might have separate parasitics files for your subblocks and a separate file for your top-level interconnect. Parasitics for chip-level timing are often read incrementally. Hierarchical parasitics for each instance are read separately and stitched with top-level parasitics. The recommended flow for reading multiple hierarchical incremental parasitics files is as follows:

```
read_parasitics A.sbpf -increment -path [all_instances -hier BLKA]/
read_parasitics B.sbpf -increment
read_parasitics C.sbpf -increment
read_parasitics D.sbpf -increment
read_parasitics chip_file_name -increment
report_annotated_parasitics -check
```

The `-increment` option enables the `read_parasitics` command to perform additional processing that allows PrimeTime to stitch together the parasitics. By default, the command does a check for incomplete annotated nets on the nets it annotates. It then executes the `report_annotated parasitics -check` command. Use the `-path` option to take a list of instances so that multiple instantiations of the same block level design can all be annotated together. This can significantly reduce both the runtime and memory required for reading parasitics in PrimeTime. By executing the commands in the recommended flow above and explicitly using the `report_annotated_parasitics -check` command, all of the parasitics

files are read in, then the entire design is checked to confirm and report the successful annotation of all nets. The report contains all of the nets in the design, not just the nets that are annotated for the last command.

Some messages issued during `read_parasitics`, `report_annotated_parasitics -check`, `read_sdf`, and an implicit or explicit `update_timing` command have a default limit each time the command is invoked. A summary of the messages affected and other useful information is stored in the log file and also printed at the end of the PrimeTime session. The `sh_message_limit` variable controls the default message limit and the `sh_limited_messages` variable controls the messages affected by this feature. For more information about the commands and variables, see the specific man page.

---

## Scaling Parasitic Values

You can scale parasitic values that have been read into the design. There are three separate scaling factors for resistors, ground capacitors, and (for PrimeTime SI users) coupling capacitors. To scale the parasitics, you use the `scale_parasitics` command.

Each factor is multiplied against all back-annotated values of the specified type throughout the design. A factor greater than 1.0 increases the parasitic component values, whereas a factor less than 1.0 decreases the parasitic component values. Factors must be greater than or equal to 0.0.

The `scale_parasitics` command operates immediately to modify the parasitics in memory. If you use the `scale_parasitics` command more than once in a session, the factor is applied to the scaled values, not the original values read from the parasitic data file. If you write the design parasitics with the `write_parasitics` command, the scaled values (not original values) are written.

---

## Net-Specific Parasitic Scaling

By using the `scale_parasitics` command with a list of nets, you can apply different scaling factors to individual nets instead of using a global scaling factor. You can scale resistance, ground capacitance, and coupling capacitance values.

The `report_scale_parasitics` command allows you to see a report of the scale parasitics. The `reset_scale_parasitics` command allows you to reset the parasitics to the global value. You can inspect the changes to the scaled parasitics using the `report_annotated_parasitics -list_annotated` command.

### Note:

Net-specific parasitic scaling does not incur a full timing update, only an incremental update.

## Ground-Capacitance and Resistance Scaling

You can scale all ground capacitances on a net by a given scaling factor. A scaling factor is a positive floating point number greater than zero. For example,

```
pt_shell> scale_parasitics [get_net n1] -ground 1.3
```

You can scale all resistances of a net by another scaling factor:

```
pt_shell> scale_parasitics [get_net n1] -resistance 1.22
```

If you scale the net multiple times or you scale globally, followed by net-specific scaling, the last command issued takes effect with respect to the original (not the previous value). For example, if the following sequence is executed, the ground capacitance of net n1 is scaled by 1.2, *not* 1.3 multiplied by 1.2:

```
pt_shell> scale_parasitics -ground 1.3 ## scale all nets
pt_shell> scale_parasitics [get_net n1] -ground 1.2 ## scale n1
```

## Coupling-Capacitance Scaling

You can scale all the coupling capacitances of a net by a scaling factor.

```
pt_shell> scale_parasitics -coupling 1.3 [get_net n1]
```

If you scale the two coupled nets by different factors, then the later command takes effect. For example,

```
pt_shell> scale_parasitics -coupling 1.3 [get_net n1]
pt_shell> scale_parasitics -coupling 1.2 [get_net n2]
```

The net effect is that the all the coupling of net n1 is scaled by 1.3 except the coupling to n2, and all the coupling of net n2 is scaled by 1.2 including the coupling to n1.

## Resetting Scale Parasitics

Use the `reset_scale_parasitics` command to reset the scaled nets to their original values. You can only issue this command on nets that have been previously scaled.

```
pt_shell> reset_scale_parasitics [get_nets n1]
```

The `reset_scale_parasitics` command returns the values to the original values, not necessarily the previous values. For example,

```
pt_shell> scale_parasitics -ground 1.3 ## scale all nets
pt_shell> scale_parasitics [get_net n1] -ground 1.2 ## scale n1
pt_shell> reset_scale_parasitics [get_net n1] ## Unscale n1
```

The net effect is that there is no scaling on net n1.



You can also use the `reset_scale_parasitics` command to regain original coupling values:

```
pt_shell> scale_parasitics -coupling 1.3 [get_net n1]
pt_shell> scale_parasitics -coupling 1.2 [get_net n2]
pt_shell> reset_scale_parasitics [get_net n1]
```

The net effect of this sequence of commands is that the coupling of n1 is unchanged, and all coupling of n2 except the coupling between n1 and n2 is scaled by 1.2.

## Reporting Scale Parasitics

Use the `report_scale_parasitics` command to look at the current scale factors of all scaled nets or a list of scaled nets:

```
pt_shell> report_scale_parasitics [get_nets n1]
```

| Net name | Ground cap factor | Resistance factor | Coupling factor |
|----------|-------------------|-------------------|-----------------|
| n1       | --                | 1.2               | 0.98            |

This means that net n1 is not scaled for ground capacitance: its resistance is scaled by a factor of 1.2, and its coupling capacitance is scaled by a factor of 0.98.

Note:

Be aware of the following possible scenario:

```
pt_shell> scale_parasitics -coupling 1.3 [get_net n1]
pt_shell> scale_parasitics -coupling 1.2 [get_net n2]
pt_shell> report_scale_parasitics [get_net n1]
```

This shows a coupling scale factor of 1.3 even though the coupling between net n1 and net n2 is scaled by a different factor, 1.2.

## Net-Specific Parasitic Scaling Examples

The following three examples illustrate net-specific parasitic scaling usage:

- [Single Net Scaling](#)
- [Physical Block \(Net\) Scaling](#)
- [Scaling Due to Net Separation](#)

### Single Net Scaling

The following example illustrates scaling the resistance and ground capacitance of a selected net. The output of the `report_annotated_parasitics` command without any scaling is as follows:

```

```

```

Report : annotated_parasitics
 -internal_nets
 -boundary_nets
 -list_annotated
 -max_nets 10

```

```

Design : Chip

```

```

Version: X-2005.06-Beta1-DEV

```

```

Date : Tue Mar 29 15:47:40 2005

```

```

```

```

1. ADDR[1] (driver: port ADDR[1])

```

| C in pF    | Node  | Pin/Port                             |
|------------|-------|--------------------------------------|
| 0.00842337 | 1     | ADDR[1] (driver) [+pin_cap=0pF]      |
| 0.0170247  | 2     | --                                   |
| 0.00531228 | 3     | --                                   |
| 0.00702817 | 4     | --                                   |
| 0.00487712 | 5     | --                                   |
| 0.00397756 | 6     | --                                   |
| 0.00590386 | 7     | --                                   |
| 0.0122464  | 8     | --                                   |
| 0.00170559 | 9     | --                                   |
| 0.00611351 | 10    | --                                   |
| 0.00304156 | 11    | --                                   |
| 0.00395301 | 12    | --                                   |
| 0.00607125 | 13    | --                                   |
| 0.00227354 | 14    | --                                   |
| 0.0119327  | 15    | --                                   |
| 0.00564987 | 16    | --                                   |
| 0.00749174 | 17    | --                                   |
| 0.00585252 | 18    | --                                   |
| 0.0083778  | 19    | --                                   |
| 0.0020378  | 20    | --                                   |
| 0.00111427 | 21    | U527/A1 (load) [+pin_cap=0.00103pF]  |
| 0.00159791 | 22    | U459/B1 (load) [+pin_cap=0.00112pF]  |
| 1e-06      | 23    | U2053/A1 (load) [+pin_cap=0.00075pF] |
| 1e-06      | 24    | U483/A1 (load) [+pin_cap=0.00107pF]  |
| 1e-06      | 25    | U534/A1 (load) [+pin_cap=0.00191pF]  |
| 1e-06      | 26    | U503/A1 (load) [+pin_cap=0.00189pF]  |
| 0.13201    | Total |                                      |

| CC in pF    | Local Node         | Other Node           |
|-------------|--------------------|----------------------|
| 0.00197175  | 13                 | n4250:13             |
| 0.00139652  | 17                 | n4250:6              |
| 0.00218098  | 3                  | n2124:2              |
| 0.00141374  | 1 ADDR[1] (driver) | REG_DATA[22]:2       |
| 0.000183647 | 16                 | n8376:2              |
| 0.000175112 | 6                  | n13565:2             |
| 0.000433937 | 6                  | n13599:4             |
| 0.000633856 | 8                  | n4750ASTipoNet3769:1 |
| 0.000346092 | 8                  | n1946:1              |

|             |    |                      |
|-------------|----|----------------------|
| 0.00274598  | 3  | n3117:1              |
| 0.00033236  | 17 | n7632:1              |
| 0.00067407  | 12 | n11321:1             |
| 0.000256377 | 2  | BW0_30_:2            |
| 0.000994872 | 3  | n225:2               |
| 0.000264488 | 15 | n1785:2              |
| 0.00157215  | 9  | n5870ASThfnNet347:5  |
| 0.00130062  | 2  | REG_DATA[26]:2       |
| 0.00170201  | 4  | n2248ASThfnNet432:10 |
| 0.00157756  | 14 | n2248ASThfnNet432:13 |

-----

0.0201561      Total

| R in Kohm   | Left node          | Right Node         |
|-------------|--------------------|--------------------|
| 0.198293    | 1 ADDR[1] (driver) | 2                  |
| 0.0190159   | 2                  | 8                  |
| 0.00350119  | 3                  | 11                 |
| 0.027464    | 3                  | 5                  |
| 0.00235238  | 4                  | 14                 |
| 0.00688478  | 4                  | 11                 |
| 0.00561526  | 5                  | 20                 |
| 0.000852706 | 6                  | 16                 |
| 0.0239033   | 6                  | 21 U527/A1 (load)  |
| 0.0380269   | 6                  | 22 U459/B1 (load)  |
| 0.0397274   | 6                  | 12                 |
| 0.0257463   | 7                  | 9                  |
| 0.0313717   | 7                  | 12                 |
| 0.00521104  | 8                  | 19                 |
| 0.00360633  | 9                  | 10                 |
| 0.0152421   | 10                 | 13                 |
| 0.0923591   | 12                 | 21 U527/A1 (load)  |
| 0.146931    | 12                 | 22 U459/B1 (load)  |
| 0.00413988  | 13                 | 17                 |
| 0.00264083  | 14                 | 17                 |
| 0.0060443   | 15                 | 18                 |
| 0.00321024  | 15                 | 19                 |
| 0.0329064   | 16                 | 23 U2053/A1 (load) |
| 0.0285095   | 16                 | 24 U483/A1 (load)  |
| 0.0187162   | 16                 | 25 U534/A1 (load)  |
| 0.0200831   | 16                 | 26 U503/A1 (load)  |
| 0.00884356  | 18                 | 20                 |
| 0.0230755   | 26 U503/A1 (load)  | 25 U534/A1 (load)  |
| 0.0667172   | 21 U527/A1 (load)  | 22 U459/B1 (load)  |
| 0.00850544  | 23 U2053/A1 (load) | 24 U483/A1 (load)  |

-----

0.909497      Total

| Net Type         | Total | Lumped | RC pi | RC network | Coupled network | Not Annotated |
|------------------|-------|--------|-------|------------|-----------------|---------------|
| Internal nets    | 0     | 0      | 0     | 0          | 0               | 0             |
| -Driverless nets |       | 0      | 0     | 0          | 0               | 0             |

|                    |   |   |   |   |   |   |
|--------------------|---|---|---|---|---|---|
| Boundary/port nets | 1 | 0 | 0 | 0 | 1 | 0 |
| - Driverless nets  |   | 0 | 0 | 0 | 0 | 0 |
|                    | 1 | 0 | 0 | 0 | 1 | 0 |

After the parasitics are scaled for resistance and ground capacitance, the output of the `report_annotated_parasitics` command would be as follows:

```

1

Report : annotated_parasitics
 -internal_nets
 -boundary_nets
 -list_annotated
 -max_nets 10
Design : Chip
Version: X-2005.06-Beta1-DEV
Date : Tue Mar 29 15:48:57 2005

1. ADDR[1] (driver: port ADDR[1])
 C in pF Node Pin/Port

0.0168467 1 ADDR[1] (driver) [+pin_cap=0pF]
0.0340494 2 --
0.0106246 3 --
0.0140563 4 --
0.00975424 5 --
0.00795512 6 --
0.0118077 7 --
0.0244928 8 --
0.00341118 9 --
0.012227 10 --
0.00608312 11 --
0.00790602 12 --
0.0121425 13 --
0.00454708 14 --
0.0238654 15 --
0.0112997 16 --
0.0149835 17 --
0.011705 18 --
0.0167556 19 --
0.0040756 20 --
0.00222854 21 U527/A1 (load) [+pin_cap=0.00103pF]
0.00319582 22 U459/B1 (load) [+pin_cap=0.00112pF]
2e-06 23 U2053/A1 (load) [+pin_cap=0.00075pF]
2e-06 24 U483/A1 (load) [+pin_cap=0.00107pF]
2e-06 25 U534/A1 (load) [+pin_cap=0.00191pF]
2e-06 26 U503/A1 (load) [+pin_cap=0.00189pF]

0.264021 Total

```

| CC in pF    | Local Node         | Other Node           |
|-------------|--------------------|----------------------|
| -----       |                    |                      |
| 0.00197175  | 13                 | n4250:13             |
| 0.00139652  | 17                 | n4250:6              |
| 0.00218098  | 3                  | n2124:2              |
| 0.00141374  | 1 ADDR[1] (driver) | REG_DATA[22]:2       |
| 0.000183647 | 16                 | n8376:2              |
| 0.000175112 | 6                  | n13565:2             |
| 0.000433937 | 6                  | n13599:4             |
| 0.000633856 | 8                  | n4750ASTipoNet3769:1 |
| 0.000346092 | 8                  | n1946:1              |
| 0.00274598  | 3                  | n3117:1              |
| 0.00033236  | 17                 | n7632:1              |
| 0.00067407  | 12                 | n11321:1             |
| 0.000256377 | 2                  | BW0_30_:2            |
| 0.000994872 | 3                  | n225:2               |
| 0.000264488 | 15                 | n1785:2              |
| 0.00157215  | 9                  | n5870ASThfnNet347:5  |
| 0.00130062  | 2                  | REG_DATA[26]:2       |
| 0.00170201  | 4                  | n2248ASThfnNet432:10 |
| 0.00157756  | 14                 | n2248ASThfnNet432:13 |
| -----       |                    |                      |
| 0.0201561   | Total              |                      |
| -----       |                    |                      |
| R in Kohm   | Left node          | Right Node           |
| -----       |                    |                      |
| 0.396586    | 1 ADDR[1] (driver) | 2                    |
| 0.0380318   | 2                  | 8                    |
| 0.00700238  | 3                  | 11                   |
| 0.054928    | 3                  | 5                    |
| 0.00470476  | 4                  | 14                   |
| 0.0137696   | 4                  | 11                   |
| 0.0112305   | 5                  | 20                   |
| 0.00170541  | 6                  | 16                   |
| 0.0478066   | 6                  | 21 U527/A1 (load)    |
| 0.0760538   | 6                  | 22 U459/B1 (load)    |
| 0.0794548   | 6                  | 12                   |
| 0.0514926   | 7                  | 9                    |
| 0.0627434   | 7                  | 12                   |
| 0.0104221   | 8                  | 19                   |
| 0.00721266  | 9                  | 10                   |
| 0.0304842   | 10                 | 13                   |
| 0.184718    | 12                 | 21 U527/A1 (load)    |
| 0.293862    | 12                 | 22 U459/B1 (load)    |
| 0.00827976  | 13                 | 17                   |
| 0.00528166  | 14                 | 17                   |
| 0.0120886   | 15                 | 18                   |
| 0.00642048  | 15                 | 19                   |
| 0.0658128   | 16                 | 23 U2053/A1 (load)   |
| 0.057019    | 16                 | 24 U483/A1 (load)    |
| 0.0374324   | 16                 | 25 U534/A1 (load)    |
| 0.0401662   | 16                 | 26 U503/A1 (load)    |

| 0.0176871          | 18    |                 |       | 20         |                 |               |  |
|--------------------|-------|-----------------|-------|------------|-----------------|---------------|--|
| 0.046151           | 26    | U503/A1 (load)  |       | 25         | U534/A1 (load)  |               |  |
| 0.133434           | 21    | U527/A1 (load)  |       | 22         | U459/B1 (load)  |               |  |
| 0.0170109          | 23    | U2053/A1 (load) |       | 24         | U483/A1 (load)  |               |  |
| -----              |       |                 |       |            |                 |               |  |
| 1.81899            | Total |                 |       |            |                 |               |  |
| Net Type           | Total | Lumped          | RC pi | RC network | Coupled network | Not Annotated |  |
| -----              |       |                 |       |            |                 |               |  |
| Internal nets      | 0     | 0               | 0     | 0          | 0               | 0             |  |
| -Driverless nets   |       | 0               | 0     | 0          | 0               | 0             |  |
| -----              |       |                 |       |            |                 |               |  |
| Boundary/port nets | 1     | 0               | 0     | 0          | 1               | 0             |  |
| - Driverless nets  |       | 0               | 0     | 0          | 0               | 0             |  |
| -----              |       |                 |       |            |                 |               |  |
|                    | 1     | 0               | 0     | 0          | 1               | 0             |  |

Note that only the resistance and the ground capacitance have been scaled by 2. All delay calculation and PrimeTime SI analysis would be updated by the new values of these parasitics.

### Physical Block (Net) Scaling

This example illustrates how you can scale each net in a physical block in your design. Foundries can provide different scaling values for different conditions of a block (process, temperature, or voltage), and the use of parasitics scaling can reflect this variation.

To use this methodology, link the top level of the design (Chip, in this example). You can then switch to the physical block of interest by using the following:

```
current_instance A

set gnets [get_net *] # This will produce a collection of
 # nets for block A.

scale_parasitics -ground 1.1 -resis 1.1 -coupling 1.1 $gnets

current_instance Chip

...

Proceed with the analysis
```

Each net in block A, is scaled as specified by the `scale_parasitics` command. Keep in mind that nets that cross hierarchies are also scaled. You can inspect the changes to the parasitics using the `report_annotated_parasitics -list_annotated` command:

### Scaling Due to Net Separation

PrimeTime has the capability to scale the coupling capacitance due to net separation. By spacing nets, you can reduce the coupling capacitance between the nets. This can be reflected using the `scale_parasitics` command. For example,

```
pt_shell> scale_parasitics -coupling 0.75 [get_net Net1]
```

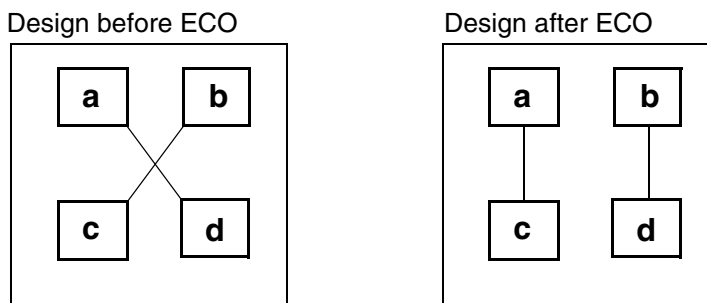
Each of the coupling capacitances for Net1 are scaled. Therefore, in this case both capacitors, from Net1 to Net2, and from Net2 to Net1, are scaled.

---

## Incremental Timing Analysis

PrimeTime can perform an incremental update on a design that is already analyzed when only a small number of nets are affected. The maximum number of nets that can be affected without requiring a full timing update depends upon the total design size. For example, if you have a full-chip annotated file and an engineering change order (ECO) annotated file, and you want to analyze the impact of your ECO, you can run an analysis with the `report_timing` command on a full-chip annotated file, and then repeat the analysis with just the ECO changes applied. See [Figure 9-7](#).

*Figure 9-7 Incremental Update Before and After Engineering Change*



To set the annotation on most—or all—nets, enter

```
pt_shell> read_parasitics full_chip.spef
pt_shell> report_timing
```

To override the annotations on a small number of nets, enter

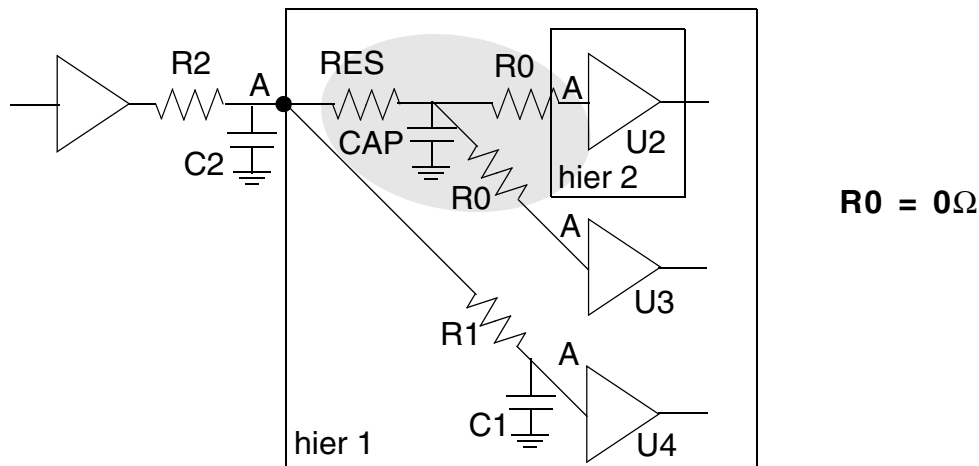
```
pt_shell> read_parasitics eco.spef
pt_shell> report_timing
```

## Incomplete Annotated Parasitics

PrimeTime can complete parasitics on nets that have incomplete parasitics. The following conditions apply to RC network completion:

- The nets that have an RC network must be back-annotated from an SPEF file.
- PrimeTime cannot complete effective capacitance calculation for a net that has incomplete parasitics. Instead, PrimeTime reverts to using wire load models for delay calculation on these nets.
- Partial parasitics can be completed on a net only if all the missing segments are between two pins (either boundary pins or leaf pins). As shown in [Figure 9-8](#), the missing segment is completed with a fanout of two.

*Figure 9-8 Missing Segment Completed Using the Fanout of Two*



The shaded area shown in [Figure 9-8](#) is completed when you use the `complete_net_parasitics` command or the `-complete_with` option of the `read_parasitics` command.

## Selecting a Wire Load Model for Incomplete Nets

Once the missing segments are identified, PrimeTime selects the wire load model to complete the missing parts of the net as follows:

- The wire load mode is taken from the top-level hierarchy. If the wire load mode does not exist, the default from the main library is taken.
- If the wire load mode is 'enclosed,' the wire load for the missing segment is the hierarchy that encloses the missing segment.



- If the wire load mode is 'top,' the wire load for the top level of hierarchy is used.
- If the wire load model does not exist on the enclosing hierarchy, the wire load model of the parent hierarchy is taken. If the parent hierarchy does not exist, the wire load model of the parent of the parent is used, this process continues until the top-level hierarchy is reached.
- The default wire load model from the main library is used if the wire load model cannot be obtained from the previous steps.
- Zero resistance and capacitance are used if no wire load model could be obtained using the `-complete_with wlm` option.

## Completing Missing Segments on the Net

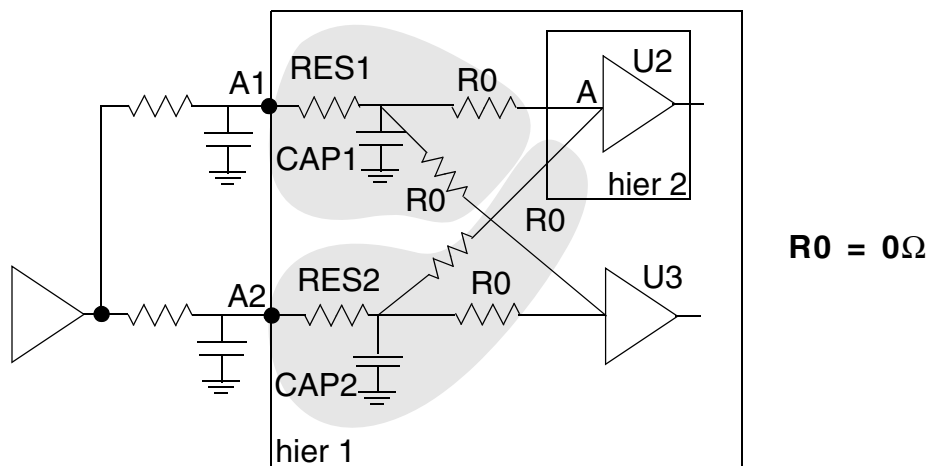
PrimeTime completes multiple pin to pin segments with a single RC pair based on the value of the option `-complete_with`, which can be `zero` or `wlm` (wire load model).

Here is a command sequence that completes the missing segments:

```
pt_shell> read_ddc design.ddc
pt_shell> read_parasitics incomplete_parasitics.spef
pt_shell> complete_net_parasitics -complete_with wlm
```

As shown in [Figure 9-9](#), if the `-complete_with wlm` option is used, the resistance (RES) and capacitance (CAP) values are estimated based on the wire load model. If `-complete_with zero` is used, the values assigned to RES and CAP are zero. In both cases, the resistance that connects the net to the loads is 0 ohms.

Figure 9-9 Multidrive Segments



You can see how networks get completed by comparing the attribute `rc_network` defined for the incomplete nets before and after completion.

After the segment is completed, it cannot be undone. For this reason, if you are using multiple `read_parasitics` commands with the `-increment` option, be sure to use the `-complete_with` option only with the very last `read_parasitics` command. Otherwise, PrimeTime completes the network before you have finished reading in all of the parasitic data files. To cancel the effects of all `read_parasitics` and `complete_net_parasitics` commands, use the `remove_annotated_parasitics` command.

Do not use the `complete_net_parasitics` command when you have parasitics with errors. However, you can manually fix the SPEF file and reread them. Use the `complete_net_parasitics` command only when the following assumption holds true: The relevant portion of a net's delay is already annotated with detailed parasitics, and you want PrimeTime to complete the remaining, less significant portion of the net with zero or wire load model based resistance and capacitance.

## Reporting Annotated Parasitics

Parasitic data files can be large (100 MB or larger) and can contain many parasitics. Use the `report_annotated_parasitics` command after reading an SPEF or RSPF file to verify that PrimeTime back-annotated all cell drivers in the design. For more information, see the `report_annotated_parasitics` man page.

To create a report for annotated parasitic data files, use the `report_annotated_parasitics -check` command

The `-check` option causes PrimeTime to verify that all RC networks are complete. PrimeTime displays a report similar to this:

```

Report : annotated_parasitics
 -check
Design : top_rc1
Version: 2000.11
Date : Thur Nov 16 14:22:37 2000

```

| Pin type           | Total | RC pi | RC network | Not Annotated |
|--------------------|-------|-------|------------|---------------|
| internal net drive | 23649 | 0     | 23649      | 0             |
| design input port  | 34    | 0     | 34         | 0             |
|                    | 23683 | 0     | 23683      | 0             |

---

## Removing Annotated Parasitics

The `remove_annotated_parasitics` command removes annotated parasitics from nets. You can remove annotated parasitics from all nets or just a specified set of nets. The `reset_design` command also removes all parasitics read and annotated using the `read_parasitics` command. For more information, see the specific man page.



# 10

## Delay Calculation With Detailed Parasitics

---

This chapter describes how PrimeTime calculates delays in the presence of detailed parasitics and the types of models used to calculate these delays.

This chapter contains the following sections:

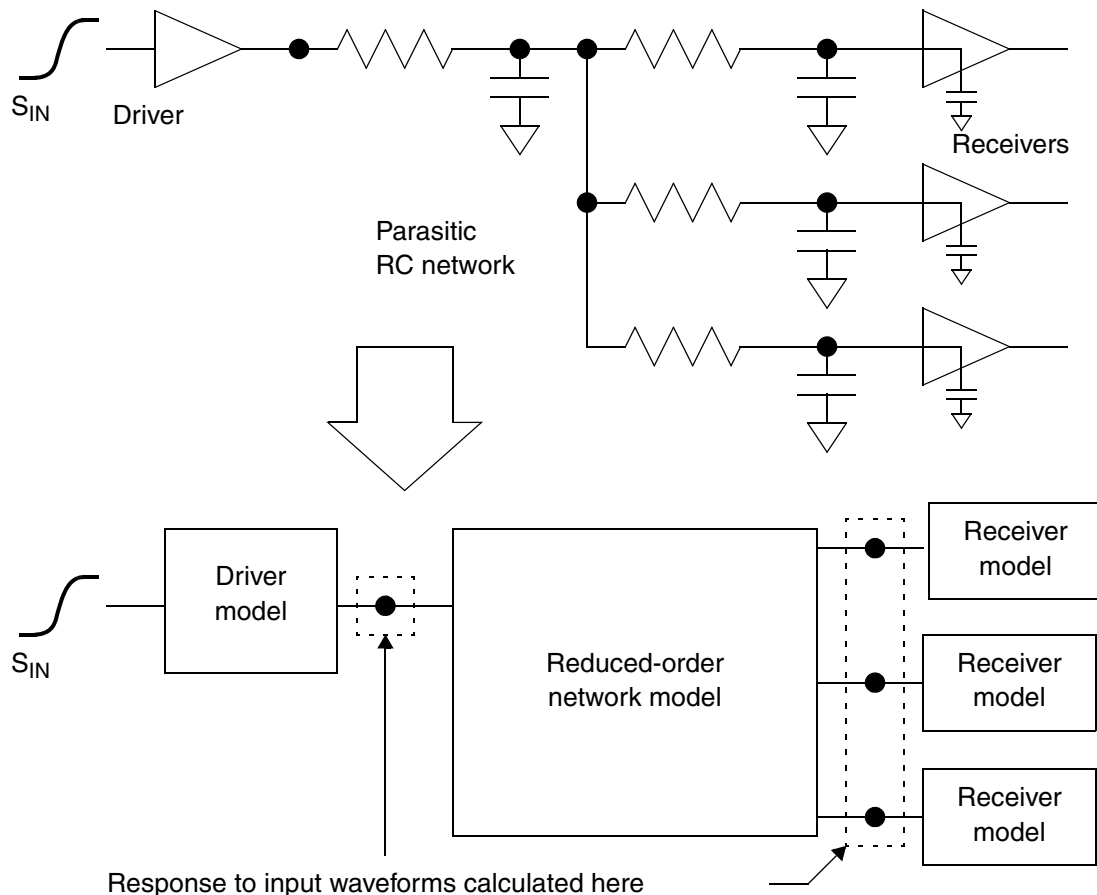
- [Overview of Delay Calculation](#)
- [Nonlinear Delay Models \(NLDM\)](#)
- [CCS Timing Models](#)
- [Scaling With CCS Timing Libraries](#)

## Overview of Delay Calculation

To perform static timing analysis, PrimeTime must accurately calculate the delay and slew (transition time) at each stage of each timing path. A stage consists of a driving cell, the annotated RC network at the output of the cell, and the capacitive load of the network load pins. The goal is to compute the response at the driver output and at the network load pins, given the input slew or waveform at the driver input, using the least amount of runtime necessary to get accurate results.

To perform stage delay calculation accurately and efficiently, PrimeTime uses models to represent the driver, RC network, and capacitive loads on the net. See [Figure 10-1](#). An ideal model would produce exactly the same delays and slews as a SPICE simulation at the output of the driver and at the input of each receiver.

*Figure 10-1 Models Used to Calculate Stage Delays and Slews*



The driver model is intended to reproduce the response of the driving cell's underlying transistor circuitry when connected to an arbitrary RC network, given a specific input slew.

The reduced-order network model is a simplified representation of the full annotated network that has nearly the same response characteristics as the original network. PrimeTime uses the Arnoldi reduction method to create this model.

The receiver model is intended to represent the complex input capacitance characteristics of a cell input pin, including the effects of rise/fall direction, the slew at the pin, the receiver output load, the state of the cell, and the voltage and temperature conditions.

PrimeTime supports two types of driver models for RC delay calculation, known as basic and advanced. When the `rc_driver_model_mode` and `rc_receiver_model_mode` variables are set to `basic`, RC delay calculation uses only the driver and receiver models derived from the more basic nonlinear delay model (NLDM) present in the cell libraries. When the variables are set to `advanced`, RC delay calculation uses the newer, more advanced Composite Current Source (CCS) timing model driver and receiver models, if data for those models are present in the cell libraries; otherwise, the basic models are used. The default setting for both variables is `advanced`.

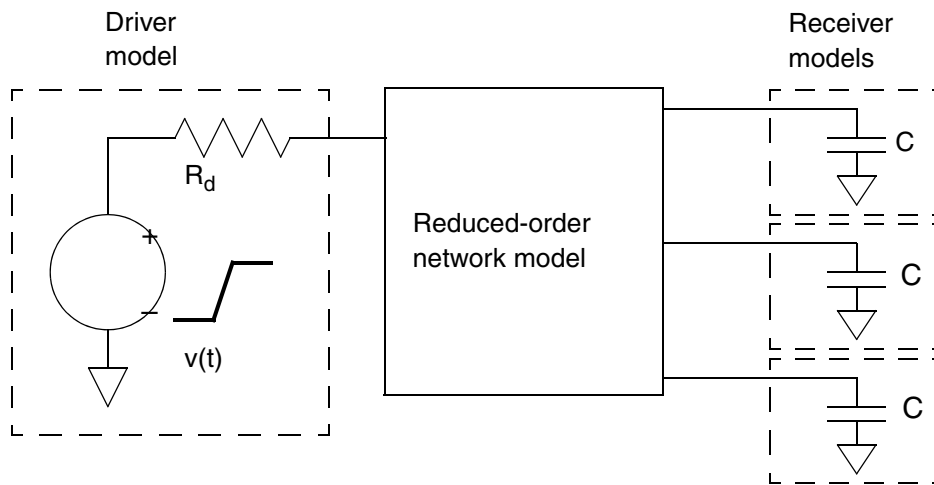
The advanced CCS timing model has many advantages, one of which is the solution to the problem described by the RC-009 warning message. This warning occurs when the drive resistance of the driver model is much less than the network impedance to ground. The CCS timing model is also better at handling the Miller Effect, dynamic IR drop, and multivoltage analysis.

---

## Nonlinear Delay Models (NLDM)

The NLDM is the earlier, established method of representing the driver and receiver of a path stage. The driver model uses a linear voltage ramp in series with a resistor (a Thevenin model), as shown in [Figure 10-2 on page 10-4](#). The resistor helps smooth out the voltage ramp so that the resulting driver waveform is similar to the curvature of the actual driver driving the RC network.

Figure 10-2 NLDM Driver and Receiver Models



The driver model has three model parameters: the drive resistance  $R_d$ , the ramp start time  $t_z$ , and the ramp duration delta  $t$ . PrimeTime chooses parameter values to match the output waveforms as closely as possible. It builds a different simplified driver model for each gate timing arc (for example, from U1/A to U1/Z) and for each sense (for example, rising edge).

When the drive resistor is much less than the impedance of the network to ground, the smoothing effect is reduced, potentially reducing the accuracy of RC delay calculation. When this condition occurs, PrimeTime adjusts the drive resistance to improve accuracy and issues an RC-009 warning. For more information, see the man page on the RC-009 warning.

The NLDM receiver model is a capacitor that represents the load capacitance of the receiver input. A different capacitance value may apply to different conditions such as rising/falling transitions or min/max timing analysis. A single capacitance value, however, applies to a given timing check, which does not support accurate modeling of the Miller Effect.

**Note:**

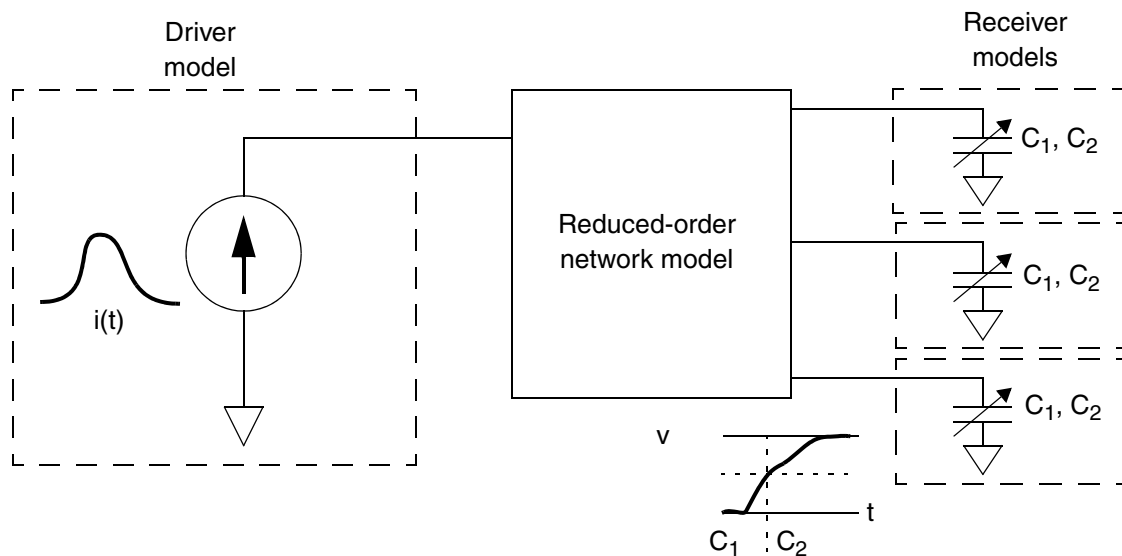
The Miller Effect is the effective change in capacitance between transistor terminals that occurs with a change in voltage across those terminals.



## CCS Timing Models

With the advent of smaller nanometer technologies, the CCS timing approach of modeling cell behavior has been developed to address the effects of deep submicron processes. The driver model uses a time-varying current source, as shown in [Figure 10-3](#). The advantage of this driver model is its ability to handle high-impedance nets and other non-monotonic behavior accurately.

*Figure 10-3 CCS Timing Driver and Receiver Models*



The CCS timing receiver model uses two different capacitor values rather than a single lumped capacitance. The first capacitance is used as the load up to the input delay threshold. When the input waveform reaches this threshold, the load is dynamically adjusted to the second capacitance value. This model provides a much better approximation of loading effects in the presence of the Miller Effect.

In some cases, different input signals can affect the input capacitance of the receiver. Conditional pin-based models are used in the library to describe the pin capacitance for different input signals. If there are conditional pin-based receiver models in the library, PrimeTime considers all receiver models and chooses the worst among the enabled pin-based and arc-based receiver models for the analysis.

In PrimeTime, the CCS timing analysis requires detailed parasitics. It uses library information in the following order for delay calculation with detailed parasitics:

1. CCS timing driver and receiver models, if both are available.
2. CCS timing driver model, if available, and lumped pin capacitance for the receiving cells.
3. NLDM delay and transition tables and pin capacitance for the receiving cells. (Use a CCS timing receiver model only with a CCS timing driver model.)

After a timing update, to determine whether CCS timing data was used for delay calculation, you can use the `report_delay_calculation` command. In the command, specify the cell or net by using the `-from` and `-to` options, or specify a timing arc using the `-of_objects` option. The following examples show CCS timing driver and receiver model data used for a cell and net delay calculation.

```
pt_shell> report_delay_calculation -from cell1/A -to cell1/Z
.....
arc sense: positive_unate
arc type: cell

Calculation Rise Rise Fall Fall Slew Rail
Thresholds: Delay Slew Delay Slew Derate Voltage Temp.

 from-pin 50 30->70 50 70->30 0.400 1.100 125.0
 to-pin 50 30->70 50 70->30 0.400 1.100 125.0
```

RC network on pin 'cell1/Z' :

```

Number of elements = 8 Capacitances + 7 Resistances
Total capacitance = 0.03062 pF
Total capacitance = 0.03062 (in library unit)
Total resistance = 0.017983 Kohm
```

Advanced driver-modeling used for rise and fall.

```

 Rise Fall
Input transition time = 0.100000 0.100000 (in library unit)
Effective capacitance = 0.002625 0.002800 (in pF)
Effective capacitance = 0.002625 0.002800 (in library unit)
Output transition time = 0.060388 0.047040 (in library unit)
Cell delay = 0.050937 0.045125 (in library unit)
```

```
pt_shell> report_delay_calculation -from [get_pins cell/Z] \
 -to [get_pins receiver/A]
...
From pin: cell/Z
To pin: receiver/A
Main Library Units: 1ns 1pF 1kOhm
arc sense: unate
arc type: net
Calculation Rise Rise Fall Fall Slew Rail
Thresholds: Delay Slew Delay Slew Derate Voltage Temp.

```

```

from-pin 50 30->70 50 70->30 1.000 0.900 125.0
to-pin 50 30->70 50 70->30 1.000 0.900 125.0

```

RC network on pin 'cell/Z' :

```

Number of elements = 8 Capacitances + 7 Resistances
Total capacitance = 0.003062 pF
Total capacitance = 0.003062 (in library unit)
Total resistance = 0.017983 Kohm
Advanced receiver-modeling used for rise and fall.

```

Advanced Receiver Model

|                            | Rise         | Fall     |                   |
|----------------------------|--------------|----------|-------------------|
| Receiver model capacitance | 1 = 0.001966 | 0.001888 | (in library unit) |
| Receiver model capacitance | 2 = 0.002328 | 0.002160 | (in library unit) |

|                          | Rise       | Fall      |                   |
|--------------------------|------------|-----------|-------------------|
| Net delay                | = 0.000024 | 0.000064  | (in library unit) |
| Transition time          | = 0.059192 | 0.037599  | (in library unit) |
| From_pin transition time | = 0.059078 | 0.037666  | (in library unit) |
| To_pin transition time   | = 0.059192 | 0.037599  | (in library unit) |
| Net slew degradation     | = 0.000114 | -0.000067 | (in library unit) |

## Pin Capacitance Reporting

Timing modeling with CCS consists of a driver model and a receiver model. As the input capacitance of a cell varies during the input signal transition, the CCS receiver model uses two capacitances, C1 and C2, to model this variation and guarantee accuracy. C1 and C2 correspond to the input capacitances before and after the delay trip point, respectively. Typically, the values of C1 and C2 are functions of input slew and output load. Therefore, it is possible that the values of C1 and C2 are larger than the library pin capacitance that is derived from CCS receiver models.

By default, the CCS receiver model information is used to compute the pin capacitance behaviors reported for load pins. CCS receiver models are used to check for max\_capacitance DRC violations, which ensures that all possible load extrapolations reported by RC-011 can be reported using the `report_constraint` command. The actual total capacitance is defined as the sum of wire capacitance and the maximum across each pin's (C1, C2) capacitance.

The following commands reflect this CCS pin capacitance reporting behavior:

- `report_attribute`
- `report_constraint -min_capacitance` and `-max_capacitance`
- `report_delay_calculation`
- `report_net`

- `report_timing -capacitance`

The following attributes reflect this CCS pin capacitance reporting behavior:

- `total_ccs_capacitance_max_fall`
- `total_ccs_capacitance_max_rise`
- `total_ccs_capacitance_min_fall`
- `total_ccs_capacitance_min_rise`

This reporting behavior is controlled by the

`report_capacitance_use_ccs_receiver_model` variable, which has a default of `true`. When you set this variable to `false`, the previous method of reporting library-derived lumped pin capacitances is used.

**Note:**

There might be inconsistencies with reporting capacitance in PrimeTime, IC Compiler, and Design Compiler. In PrimeTime PX, there might be a mismatch of capacitance values between the reports displayed when using the `report_delay_calculation` and `report_power_calculation` commands.

---

## Guidelines to Address the CCS Extrapolation Warning Message (RC-011)

Extrapolation warning messages (RC-011) occur when RC delay-calculation is attempted using a slew or load that is much larger than the maximum library slew or load indices of a cell or pin. Large extrapolations can cause inaccurate results.

The following sections describe the recommendations to address the driver extrapolation and receiver load extrapolations:

- [Guidelines for Characterizing Design Rule Constraints](#)
- [Guidelines for Fixing RC-011 Warning Messages](#)

## Guidelines for Characterizing Design Rule Constraints

The `max_transition` pin attributes are normally present on the input and output pins of library cells. For input pins, the `max_transition` attribute value should not exceed the maximum slew index in the NLDM and CCS driver and CCS receiver `capacitance2` tables. The lowest value of the maximum slew index between the NLDM and CCS tables should be used as a reference. The tables used as reference are for the rising and falling timing arcs from the relevant input pin for which the `max_transition` attribute is being characterized. You should take both the arc-based and pin-based tables into account.

The `max_capacitance` pin attributes are normally present on the output pins of library cells. For output pins, the `max_capacitance` attribute value should not exceed the maximum load index in the NLDM and CCS driver as well as CCS receiver\_capacitance1 and receiver\_capacitance2 tables. You should use the lowest value of the maximum load index between the NLDM and CCS tables as a reference. The tables used as reference are for the rising and falling timing arcs to the relevant output pin for which the `max_capacitance` attribute is being characterized. You should take both the arc-based and pin-based tables into account.

## Guidelines for Fixing RC-011 Warning Messages

If the libraries used in the design follow the guidelines outlined in the [Guidelines to Address the CCS Extrapolation Warning Message \(RC-011\)](#) section, all RC-011 messages are addressed by fixing the `max_transition` and `max_capacitance` violations reported by the `report_constraint` command. To ensure all the `max_capacitance` violations in RC-011 are reported by the `report_constraint` command, the `report_capacitance_use_ccs_receiver_model` variable is set to its default of `true`. If the libraries are not compliant with these guidelines, you should consider RC-011 warning messages to be important. You need to address the design rule constraints in the design to fix these warnings.

---

## Scaling With CCS Timing Libraries

PrimeTime supports voltage and temperature scaling by interpolating between data in separate libraries that have been characterized at different voltage and temperature values. The delay (CCS timing driver model and receiver model) and timing constraints are scaled. In addition, scaling occurs if there is a mixture of CCS and NLDM data. Scaling between the libraries is done during runtime of the tool.

---

## Invoking Scaling

You can invoke voltage and temperature scaling by using the `define_scaling_lib_group` command. This command specifies the scaling relationships between libraries that have been characterized at different voltages and temperatures and invokes both delay and constraint scaling. You can define scaling relationships between multiple libraries as shown in the following example:

```
pt_shell> define_scaling_lib_group \
 {lib_0.9v_0C.db lib_1.1v_0C.db lib_1.3v_0C.db}
```

This command should be issued after the design has been read in. If the design is not already linked, the `define_scaling_lib_group` command automatically links the design and creates scaling relationships between the libraries in each group. If the design is already

linked, this command creates scaling relationships without an additional link. There is no restriction on the number of libraries you can specify in the command, and you can define multiple scaling groups to cover different portions of your design. Each library, however, can be part of only one scaling group.

---

## Guidelines for Scaling

When using the `define_scaling_lib_group` command:

1. Read in the design netlist before defining scaling relationships.
2. Only one library in the group should be in the link path. The remaining libraries are read in automatically during creation of the scaling group. Each library can be part of only one scaling group.
3. While creating scaling relationships, PrimeTime reports the following information:

```
Completing scaling library groups ...
Loading db file './lib/lib_0.9V_0C.db'
Loading db file './lib/lib_1.3V_0C.db'
... the scaling library groups are complete.
```

During this process, PrimeTime does consistency checking between the libraries. The cell names, pin names, and timing arcs need to be identical between libraries, and the output capacitance table indices must be the same for each timing arc.

All libraries in the scaling group must be consistent in terms of the existence of the receiver model. For example, for a specific library pin, either all libraries have a CCS receiver model or all the libraries do not have any CCS receiver model.

4. Set the supply voltages and temperature of each cell using the `create_operating_conditions` and `set_operating_conditions` commands. If you are using IEEE 1801 Unified Power Format (UPF) as part of a low-power design, you should use the `set_voltage` and `set_temperature` commands. Ensure that each specified voltage and temperature value is within the range of the applicable scaling library group.
5. If a library is removed from memory with the `remove_lib` command, the affected library group is removed without warning.

The `report_delay_calculation` command provides additional information when scaling is used for the delay calculation. It reports which transitions (rise and/or fall) used scaling and which libraries were used for delay calculation.

```
pt_shell> report_delay_calculation -from [get_pins cell/A] \
 -to [get_pins cell/Z]

...
```

```
arc sense: negative_unate
arc type: cell
```

```
RC network on pin 'cell/Z' :
```

```

Number of elements = 4 Capacitances + 3 Resistances
Total capacitance = 0.015272 pF
Total capacitance = 15.271626 (in library unit)
Total resistance = 0.027793 Kohm
```

```
Advanced driver-modeling used for rise and fall.
Scaling library arc group used for rise and fall.
Scaling libraries used for driver model :
 /remote/testcase/lib/ccs_lib_1.05.db:ccs_1.05
 /remote/testcase/lib/ccs_lib_0.85.db:ccs_0.85
```

```

 Rise Fall

Input transition time = 0.600000 0.600000 (in library unit)
Effective capacitance = 0.015272 0.015272 (in pF)
Effective capacitance = 15.271626 15.271626 (in library
unit)
Output transition time = 0.091507 0.088209 (in library
unit)
Cell delay = 0.031275 0.112788 (in library unit)
```

The `report_lib_groups` command can be used to find all libraries with scaling relationships in the design. For example,

```
pt_shell> report_lib_groups -scaling -show {voltage temperature}
```

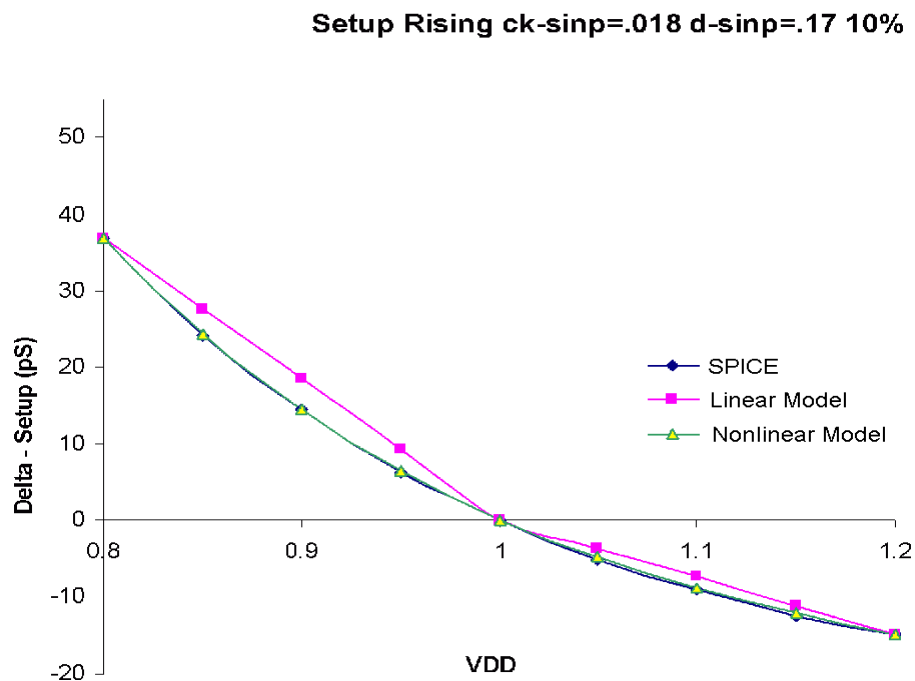
```
...
```

| Group   | Library  | Temperature | Voltage |
|---------|----------|-------------|---------|
| -----   |          |             |         |
| Group 1 |          |             |         |
|         | lib_0.85 | 125.00      | 0.85    |
|         | lib_1.05 | 125.00      | 1.05    |

## Scaling Interpolation for Constraints

For setup, hold, recovery, removal, minimum pulse width, and minimum period constraints, PrimeTime uses a nonlinear interpolation method for voltage variation. Figure 10-4 shows a comparison of SPICE data, linear interpolation, and nonlinear interpolation for a D flip-flop setup constraint in the voltage range between 0.8 and 1.2 volts. The nonlinear interpolation method used by PrimeTime produces more accurate results than linear interpolation.

Figure 10-4 Nonlinear Interpolation of Timing Constraint



## Scaling of Design Rule Constraints

If scaling has occurred, the `report_constraint` command can interpolate the design rule constraint values between the scaling libraries that were used for calculation on the corresponding pin. Design rule constraint support for composite current source (CCS) scaling eliminates any possibility of missed design rule constraint violations.

## Scaling Support for Multirail Cells

PrimeTime provides full scaling support for multirail cells, such as level shifters. Level shifter cells operate across voltage differences, connecting driver and load pins of cells belonging to different power domains and placed in different voltage areas. Level shifter cells are



frequently used in low-power design. They are used to connect the different voltage power domains by stepping the voltage up or down, as needed. To analyze the dependencies on multiple variables, including multiple rail voltages and temperature, accurate multirail cell scaling is needed.

Data for multirail cell scaling supports timing, noise, and power data. This feature supports scaling library groups of power and ground (PG) pin-based CCS libraries, non-PG libraries, legacy rail\_connection libraries, and power domain mode commands with multirail cells; all other scaling library groups are supported using single-rail scaling.

Multirail scaling is enabled by default for timing, noise, and power analysis. PrimeTime automatically detects a multirail scenario and applies multirail scaling when there are enough libraries in the scaling library group to support it. If there are not enough libraries in the scaling library group, PrimeTime performs single-rail scaling and issues an SLG-211 warning message.

**Note:**

The multirail scaling feature is currently not applicable for variation analysis using PrimeTime VX. Enable variation analysis by setting the `variation_enable_analysis` variable to `true` before using the `define_scaling_lib_group` command to define libraries supporting voltage and temperature scaling.

To set up multirail scaling, you should use the following flow:

1. Use the `define_scaling_lib_group` command to set up scaling library groups to support voltage and temperature scaling.

Ensure that there is at least one scaling library group for each scaling scenario. Treat scaling with different dimensions as different scaling scenarios. For example, you should have at least one scaling library group for regular one-rail cells and another for two-rail level shifters that require scaling in more dimensions. For example,

```
pt_shell> define_scaling_lib_group {std_lib_0.9V.db std_lib_1.1V.db}
pt_shell> define_scaling_lib_group {LS_lib_0.9V_0.9V.db \
LS_lib_0.9V_1.1V.db \
LS_lib_1.1V_0.9V.db \
LS_lib_1.1V_1.1V.db }
```

**Note:**

If multirail scaling fails, the timing analysis falls back to single-rail scaling.

2. Put cells that require scaling in different dimensions into different libraries and those libraries must be put into different scaling library groups. For example, two-rail level shifter cells require at least eight libraries to perform voltage and temperature scaling if you are using the on-the-grid formation. All other one-rail cells that do not need to consider underdrive and overdrive can be put in one library, and only four libraries are needed to perform simultaneous voltage and temperature scaling of these one-rail cells. If this condition is not met, error message SLG-202 is issued.

3. Set the design instance operating conditions using the `set_voltage`, `set_rail_voltage`, `set_operating_conditions`, and `set_temperature` commands.
4. Use the `report_lib_groups` command to generate a report that displays the library groups. This report has been enhanced to display all of the voltage rail information in the library scaling groups if multirail scaling is supported for that library group. For example,

```
pt_shell> report_lib_groups -scaling -show {voltage temp process}

```

```
Report : lib_groups
 -scaling
 -show
```

```
Design : cpu
Version: E-2010.12
Date : Tues Dec 21 10:37:05 2010

```

| Group   | Library                   | Temperature | Voltage            | Process |
|---------|---------------------------|-------------|--------------------|---------|
| -----   |                           |             |                    |         |
| Group 1 |                           |             |                    |         |
|         | tcbn65lpwc_ccs            | 125.00      | 1.08               | 1.00    |
|         | tcbn65lpwc0d72_ccs        | 125.00      | 0.86               | 1.00    |
| Group 2 |                           |             |                    |         |
|         | tcbn65lplvtwc0d720d72_ccs | 125.00      | { V:0.86 VL:0.86 } | 1.00    |
|         | tcbn65lplvtwc0d720d9_ccs  | 125.00      | { V:1.08 VL:0.86 } | 1.00    |
|         | tcbn65lplvtwc0d90d72_ccs  | 125.00      | { VL:0.86 V:1.08 } | 1.00    |
|         | tcbn65lplvtwc0d90d9_ccs   | 125.00      | { V:1.08 VL:1.08 } | 1.00    |
| Group 3 |                           |             |                    |         |
|         | tcbn65lplvtwc_ccs         | 125.00      | 1.08               | 1.00    |
|         | tcbn65lplvtwc0d72_ccs     | 125.00      | 0.86               | 1.00    |

To use the multirail scaling feature on multirail cells, you must ensure that the libraries meet the following requirements:

- Library operation conditions in a scaling group should be in on-the-grid,  $2n+1$ , or  $n+1$  type of library group formations, which are currently acceptable by the multirail scaling feature. On-the-grid library formation means the operation conditions of the libraries have to be on the grid in the Cartesian coordinate system. This requires at least  $2^n$  libraries for  $n$  dimensional scaling. In the above example, the second scaling library for two-rail level shifters is using on-the-grid library formation. The  $2n+1$  type of library formation has a nominal library in the middle. In addition, there is one plus side library and one minus side library in each dimension. Therefore, the set of libraries have a total of  $2n+1$  libraries. The  $n+1$  type of library formation is similar to the  $2n+1$  type, but it only has the plus or minus side. If the library operating condition distribution is improper, the SLG-209 error message is issued and scaling is not performed.
- Cells that require scaling in different dimensions are put into different libraries. For example, two-rail level shifter cells require at least eight libraries to perform voltage and temperature scaling, if you are using the on-the-grid formation. All other one-rail cells that

do not need to consider underdrive and overdrive can be put in one library, and only four libraries are needed to perform simultaneous voltage and temperature scaling of these one-rail cells. If this condition is not met, error message SLG-202 is issued.

- PrimeTime and PrimeTime PX provides the capability to convert and update library power and ground (PG) pins and non-PG pin libraries through the UPF library PG pin conversion.
- Libraries can contain timing, noise, and power information. Multirail scaling is applied to each of these data types of multirail cells.
- Expanded CCS libraries, CCS libraries using the base curve technology format, and NLDM-only libraries are supported.

**Note:**

Multirail scaling is not supported in PrimeTime VX. It is automatically disabled if you read in variation libraries. If you have already defined scaling groups for these multirail cells, variation libraries for multirail cells are not read into the design.

Multirail cell scaling enables you to avoid using the `link_path_per_instance` variable. This feature is capable of not only scaling all rails but also figuring out which libraries match exactly. PrimeTime automatically selects the correct library in the scaling library group if the operating condition of a design cell or pin instance exactly matches the operating condition of a library in the scaling library group. When the operating conditions of one of the libraries in the scaling library group exactly matches the operating conditions of a cell/pin instances, multirail scaling automatically picks up this library and uses it in calculations.



# 11

## Low-Power Flow Support

---

PrimeTime supports the use of the IEEE 1801 Unified Power Format (UPF) Standard for specifying the low-power features of the design. PrimeTime correctly analyzes the timing of the design in the presence of multivoltage supplies and voltage values set on specific supply nets with the `set_voltage` command.

The low-power flow support is described in the following sections:

- [Multivoltage Analysis](#)
- [UPF Commands](#)
- [Virtual Power Network](#)
- [Setting Voltage and Temperature](#)
- [Analysis With Multiple Voltages](#)
- [Multivoltage Reporting and Checking](#)
- [UPF Supply Sets](#)
- [Library PG Tcl File](#)
- [Power Domain Mode of Release Z-2007.06](#)
- [Multivoltage Method Prior to Release Z-2007.06](#)

---

## Multivoltage Analysis

PrimeTime supports timing analysis with different power supply voltages on different cells in the design. PrimeTime calculates delays and slews, PrimeTime SI calculates crosstalk delay and noise effects, and PrimeTime PX calculates power consumption based on the actual supply voltage on each supply pin of each cell.

To perform a multivoltage analysis, you must provide information about the voltage on each power supply pin of each cell in the design. There are three different ways to provide this information:

- Specify the power intent of the design using UPF commands and related PrimeTime commands, including `create_power_domain`, `create_supply_net`, `create_supply_set`, `create_supply_port`, `connect_supply_net`, and `set_voltage`. This method was introduced in the A-2007.12 release and is the preferred method.
- Use commands in the “power domain mode” of the Z-2007.06 release. The commands in this mode are `create_power_net_info`, `create_power_domain`, `connect_power_domain`, `connect_power_net_info`, `set_operating_conditions -object_list`, and `set_voltage`. To use the power domain mode from the Z-2007.06 release, you must set the `power_domains_compatibility` variable to `true`, which disables the other two methods. For more information, see [“Power Domain Mode of Release Z-2007.06” on page 11-26](#).

**Note:**

In the power domain mode, the `create_power_domain` command has a different effect from the command of the same name executed in the default UPF mode.

- Set the supply voltages on specific blocks in the design by using either the `set_operating_conditions -object_list` or `set_rail_voltage` command or both commands. Using the `set_rail_voltage` command requires a PrimeTime SI license. This method was available prior to the Z-2007.06 release and is still supported. For more information about using this method, see [“Multivoltage Method Prior to Release Z-2007.06” on page 11-28](#).

The three methods are mutually exclusive. By default, if you use any UPF commands, UPF is the voltage specification method for the PrimeTime session. If you do not use any UPF commands in the session, you can use the method available prior to the Z-2007.06 release by using either the `set_operating_conditions -object_list` or `set_rail_voltage` command or both commands.

Most of this chapter applies to the recommended UPF flow. For information that is specific to the two older flows, see the applicable sections at the end of this chapter, [“Power Domain Mode of Release Z-2007.06” on page 11-26](#) and [“Multivoltage Method Prior to Release Z-2007.06” on page 11-28](#).

Unless you are using the older method of specifying all supply voltages using `set_operating_conditions` or `set_voltage`, the library must contain power and ground (PG) pin information for each cell in the design. The Liberty syntax for specifying PG pin information in the library uses the `voltage_map` and `pg_pin` statements. The `voltage_map` statement defines the power supplies and default voltage values, whereas the `pg_pin` statements specify the power supply associated with each pin of each cell, as well as some of the power-related pin parameters. For more information about library requirements for multivoltage analysis, see the chapter called “Advanced Low Power Modeling” in the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*, which is available in the Library Compiler collection on SolvNet.

PrimeTime supports partial and no PG pin cells in the Liberty PG pin library. Partial PG pin cells are cells that are missing power pins, ground pins, or both power and ground pins. For cells that have no signal pins, PrimeTime removes these cells from the design at link time. No timing or power information is reported for these types of cells.

PrimeTime and PrimeTime PX provide the capability to automatically convert and update the library power and ground (PG) pins. With this feature, the tool creates PG pins automatically through library conversion, and the library update specifications are provided in a PG Tcl file. For more information about the PG Tcl file, see [“Library PG Tcl File” on page 11-24](#).

The second essential requirement for multivoltage analysis is Liberty library data that accurately describes timing behavior at specific voltages. PrimeTime supports voltage scaling by interpolating between data in separate libraries that have been characterized at different supply voltages. You invoke voltage and temperature scaling by using the `define_scaling_lib_group` command. Note that the behavior of PrimeTime is different from that of Design Compiler, which links the design at precise corner voltages. For more information, see the man page for the `define_scaling_lib_group` command or [“Scaling With CCS Timing Libraries” on page 10-9](#).

---

## UPF Commands

Multiple power supplies at different voltages can supply power to different domains occupying different areas of the chip. Some power domains can be selectively shut off to conserve power during periods of inactivity. Level shifter cells convert signals leaving one domain and entering another, while isolation cells supply constant signal levels at the outputs of domains that are shut down. Power-down domains can contain retention registers that can retain logic values during the power-down period. Power-switch cells, operating under the control of a power-controller block, switch the power on and off to specific domains. You can specify all of these low-power aspects of a design in the UPF language.

A standard PrimeTime license includes UPF support. No additional license is required specifically for UPF; however, power analysis requires a PrimeTime PX license and IR drop annotation requires a PrimeTime SI license.

For background information about the Synopsys low-power flow and using UPF commands in the flow, see the *Synopsys Low-Power Flow User Guide*, which is available in the PrimeTime Suite collection on SolvNet. The same user guide is also available in other product documentation collections because you can use the UPF syntax with a wide range of products, including Design Compiler, IC Compiler, Formality, MVSIM-VCS, MVRC, and PrimeRail.

PrimeTime uses UPF-specified power intent and the `set_voltage` command to determine the voltage on each power supply pin of each cell. Based on the UPF-specified power intent and `set_voltage` information, PrimeTime builds a virtual model of the power network and propagates the voltage values from UPF supply nets to the PG pins of leaf instances. Since PrimeTime does not directly read power state tables, the `set_voltage` command should be consistent with a specific state in the UPF power state table (PST) that you intend to verify.

You can enter the UPF commands at the shell prompt. You can also source these commands in a command file or with the `source` or `load_upf` command. Any loaded UPF information is removed upon relinking a design, just like timing assertions loaded from a Synopsys Design Constraints (SDC) file.

The following is a typical sequence of commands used in a PrimeTime timing analysis with UPF-specified power intent. The UPF-related power commands are highlighted in bold.

```
Read libraries, designs
...
read_lib l1.db
read_verilog d1.v
...

Read UPF file
(containing commands like the ones shown below)

create_power_domain ...
create_supply_set ...
create_supply_net ...
create_supply_port ...
create_power_switch ...
connect_supply_net ...
set_scope block1
load_upf block1_upf.tcl

Read SDC and other timing assertions
source d1.tcl

Define scaling library groups for voltage and temperature scaling
define_scaling_lib_group library_list
```



```
Read SDC and other timing assertions
...
set_voltage -object_list supply_net_name
set_voltage -cell ... -pg_pin_name ... value
(sets voltage on supply nets or IR drop on cell power pins;
PrimeTime SI license required for -cell and -pg_pin_name options)
set_temperature -object_list cell_list value

Perform timing, signal integrity analysis
report_timing
```

PrimeTime reads and uses the UPF information, but it does not modify the power domain description in any structural or functional way. Therefore, it does not write out any UPF commands with the `write_script` command and there is no `save_upf` command.

PrimeTime supports a subset of the commands and command options in the UPF 1.0 standard (available from <http://www.accellera.org>). The unsupported commands are either unrelated to the functions of PrimeTime (for example, synthesis and physical implementation) or represent capabilities that have not been implemented.

The supported power supply commands can be divided into the following categories:

- Power domain commands:

```
connect_supply_net
create_power_domain
create_power_switch
create_supply_net
create_supply_port
create_supply_set
set_domain_supply_net
```

- Isolation and retention commands:

```
set_isolation
set_isolation_control
set_port_attributes
set_design_attributes
set_retention
set_retention_control
```

- Flow commands:

```
load_upf
set_scope
set_design_top
upf_version
```

- **Related non-UPF commands:**

```
check_timing -include signal_level
check_timing -include supply_net_voltage
check_timing -include unconnected_pg_pins
get_power_domains
get_power_switches
get_supply_nets
get_supply_ports
get_supply_sets
report_power_domain
report_power_network
report_power_pin_info
report_power_switch
report_supply_net
report_supply_set
set_level_shifter_strategy
set_level_shifter_threshold
set_related_supply_net
set_temperature
set_voltage
```

Some UPF commands are essential for synthesis and other implementation tools, but they supply information that is either not used during PrimeTime analysis or is available from other sources. PrimeTime accepts these commands as valid UPF syntax, but otherwise ignores them. It does not provide man pages for these commands. The following UPF commands are ignored by PrimeTime:

```
add_port_state
add_pst_state
create_pst
map_isolation_cell
map_level_shifter_cell
map_power_switch
map_retention_cell
name_format
set_level_shifter
set_power_switch
```

The following commands are rejected by PrimeTime as unknown syntax. They are either not supported by the Synopsys UPF command subset in all Synopsys tools or they exist only in the RTL UPF.

```
add_domain_elements
bind_checker
create_hdl2upf_vct
create_upf2hdl_vct
merge_power_domains
save_upf
set_pin_related_supply
```

The following sections describe the supported UPF commands and their usage in PrimeTime. For background information or more information about using each command throughout the synthesis, implementation, and verification flow, see the UPF 1.0 specification, available as a free download from Accellera at <http://www.accellera.org>. For more information about the Synopsys low-power flow and Synopsys usage of UPF commands, including usage in PrimeTime, see the *Synopsys Low-Power Flow User Guide*.

---

## Virtual Power Network

In the PrimeTime multivoltage analysis flow, you typically read in the design netlist, source a UPF command script that defines the power supply intent, and then source an SDC script that specifies the timing constraints for analysis. From the design netlist and UPF commands, PrimeTime builds a virtual power network that supplies power to all the PG pins of all the leaf-level cells in the design netlist.

PrimeTime builds the power supply network based on UPF commands, including `create_power_domain`, `create_supply_net`, `create_supply_set`, `create_supply_port`, `set_domain_supply_net`, and `connect_supply_net`. It determines which cells belong to which power domains and traces the supply connections through the supply nets and supply ports down to the leaf level.

The `set_retention` and `set_retention_control` UPF commands determine the supply connections to the retention registers. Similarly, the `set_isolation` and `set_isolation_control` UPF commands determine the connections of the backup supply nets of the isolation cells. The `set_port_attributes` and `set_design_attributes` commands specify the attributes of the source or sink elements for the `set_isolation` command. The `set_port_attributes` command specifies information that is relevant to ports on the interface of the power domains. This information is used to determine isolation and guard requirements for the port. The `set_related_supply_net` command specifies the attributes and their values on desired cells.

The voltage of the supply port is determined in the following order of precedence, from highest to lowest:

- The `-driver_supply` or `-receiver_supply` option, along with the `-port` option, for driver supply on input ports and receiver supply on output ports
- The `-driver_supply` or `-receiver_supply` option, along with `-element` option, for driver supply on input ports and receiver supply on output ports
- The `set_related_supply_net` command
- The primary supply of the power domain of the top design

The `set_level_shifter` UPF command is not supported in PrimeTime; therefore, the `connect_supply_net` command must explicitly specify the power supply connections of the PG pins of level shifters. Design Compiler automatically generates the `connect_supply_net` commands when it inserts level shifters and writes them out with the `save_upf` command. Therefore, you do not need to create these commands yourself when you read the UPF produced by Design Compiler. To specify the behavior of signal level mismatch checking by the `check_timing -include signal_level` command, use the `set_level_shifter_strategy` and `set_level_shifter_threshold` non-UPF commands. These commands do not affect PG connectivity.

The `set_related_supply_net` UPF command associates a supply net to one or more ports of the design. Use the `object_lists` option of the `set_related_supply_net` command to specify the list of ports associated with supply nets. In the absence of this command, PrimeTime assumes that ports are supplied by the voltage of the design operating condition. Using the `-receiver_supply` and `-driver_supply` options of the `set_port_attributes` command, you can replace the `set_related_supply_net` command with the `set_port_attributes` command. You can specify the `-receiver_supply` option only on the top-level ports and design. PrimeTime PX issues a warning message if you specify ports that are not on the top level of the design hierarchy, and the attribute is ignored.

---

## Setting Voltage and Temperature

To determine the supply voltage on each PG pin of each cell in the design, PrimeTime uses the following order of precedence, from highest to lowest. The UPF port voltage has higher precedence than the non-UPF port voltage.

- The `-driver_supply` or `-receiver_supply` option, along with the `-port` option, for driver supply on input ports or receiver supply on output ports
- The `-driver_supply` or `-receiver_supply` option, along with `-element` option, for driver supply on input ports or receiver supply on output ports
- The `set_related_supply_net` command
- The primary supply of the power domain of the top design
- The `set_voltage` on a PG pin (PrimeTime SI license required)
- The `set_voltage` on a supply net
- The `set_operating_conditions` applied at the design level
- The default supply voltage in library cell definition (`voltage_map` in Liberty syntax)

By default, PrimeTime uses the supply voltages in the library cell definitions, as specified by the `voltage_map` statement in Liberty syntax for the library cell. For more information, see the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*, available in the Library Compiler collection on SolvNet.

Override the library-specified voltages by using the `set_operating_conditions` command to set the operating conditions at the design level (in other words, without using the `-object_list` option of the command). For example,

```
pt_shell> set_operating_conditions WCIND
```

The supply voltages associated with the named operating condition are specified in the library definition for the operating condition or by the `create_operating_condition` command.

Override both the library-specified and operating-condition voltage settings with the `set_voltage` command for specific supply nets or specific PG pins in the design.

The command specifies one or more supply nets (`-object_list supply_nets`) or PG pins (`-cell cell -pg_pin pg_pin`) in the design and applies a specified voltage to the listed objects. You can specify either a single voltage (`max_voltage`) or both the minimum-delay and maximum-delay voltages (`-min min_voltage` and `max_votltage`) used for analysis. Note that the `-min min_voltage` option specifies the voltage used for minimum-delay analysis, which is typically the larger voltage value. A PrimeTime SI license is required to use the `-cell` and `-pg_pin_name` options. For example, to set a voltage of 1.20 on the supply net VDD1, you would use the following command:

```
pt_shell> set_voltage 1.20 -object_list VDD1
```

To set a maximum-delay voltage of 0.91 and a minimum-delay voltage of 1.18 on the PWR pin of cell XA91, you would use the following command (PrimeTime SI license required):

```
pt_shell> set_voltage 0.91 -min 1.18 -cell XA91 -pg_pin_name PWR
```

A PAD pin is defined by having its `is_pad` attribute set to `true` in the Liberty description of the pin. PrimeTime detects ports connected to PAD cells and determines the port voltage in the following order of increasing priority:

1. The nominal voltage of the main library
2. The nominal voltage of the driving library cell
3. The design operating condition
4. The voltage of the connected pin of the PAD cell
5. The explicit port voltage, set with the `set_related_supply_net` command in UPF mode or the `set_operating_conditions -object_list port` command in non-UPF mode

Since the operating condition defines only the single voltage value, you should exercise caution when setting operating conditions on designs containing multirail cells, such as level shifters and power management cells. PrimeTime applies the operating condition voltage value only to the first library power rail. For example, the value is applied to the first nonground voltage\_map line in the .lib file. Thus, the order of the voltage\_map entries in the .lib file is important in this partial multivoltage flow.

The correct way to completely avoid this issue on multirail cells is to use a proper multivoltage flow, such as UPF, that defines all design rails using the `set_voltage` command. To verify voltage values of all rails of a multirail cell, use the `report_power_pin_info` command.

Use the `report_power_pin_info` and `report_timing -voltage` commands to display the voltage of PG and signal pins respectively. For more information, see [“Reporting Commands” on page 11-15](#) or the man page.

You can similarly specify the operating temperature for specific cells by using the `set_temperature` command, overriding the design operating condition settings. Use this command to model the effects of on-chip temperature variation. For example, you can use the following syntax to set the temperature to 130 degrees Celsius on cell XA91:

```
pt_shell> set_temperature 130 -object_list [get_cells XA91]
```

---

## Analysis With Multiple Voltages

You can provide a set of libraries characterized at a range of supply voltage and temperature values, and then scale the cell data between these libraries. This is the preferred way to provide multivoltage library data. The `define_scaling_lib_group` command specifies the list of CCS libraries that have been characterized at different voltages and temperatures, and also invokes delay and constraint scaling between those libraries. PrimeTime interpolates between the specified libraries to determine the cell behavior at the exact voltage and temperature conditions that have been set on each cell. For more information, see the man page for the `define_scaling_lib_group` command or [“Scaling With CCS Timing Libraries” on page 10-9](#).

Another way to provide multivoltage library data is to use the `link_path_per_instance` variable to direct PrimeTime to use different libraries for different cell instances. This variable, when set prior to linking the current design, overrides the default `link_path` setting

for selected leaf cell or hierarchical cell instances. The format is a list of lists. Each sublist consists of a pair of elements: a set of instances and a `link_path` specification to be used for those instances. For example,

```
set link_path {* lib1.db}
set link_path_per_instance [list
 [list {ucore} {* lib2.db}]
 [list {ucore/usubblk} {* lib3.db}]]
```

The listed instances can be hierarchical blocks that represent individual voltage domains. The corresponding libraries might be characterized at different supply voltages or under varying conditions such as body bias. If a given block matches multiple entries in the per-instance list, the more specific entry overrides the more general entry. In the example above:

- lib3.db is used to link blocks “ucore/usubblk” and below.
- lib2.db is used to link “ucore” and below (except within “ucore/subblk”).
- lib1.db is used for the remainder of the design (everything except within “ucore”).

Using either type of library data method, PrimeTime uses the scaled or library-specified supply voltage information to accurately determine the delays and slews of signals.

For post-layout analysis using detailed annotated parasitics, PrimeTime determines the delays and slews using analog waveforms. For pre-layout analysis (in the absence of detailed annotated parasitics), PrimeTime performs geometric-like scaling of the transition times and net delays, taking into account the respective voltage swings of the driver and load, and the logic thresholds. In either case, PrimeTime reports transition times in terms of local-library thresholds and local voltages.

The use of local trip points and voltages can cause an apparent improvement in transition time along nets in a path. For example, a driver cell might have a transition time of 1.0 ns measured between 20 percent and 80 percent of VDD, while the load on the same net has a transition time of 0.67 ns measured between 30 percent and 70 percent of VDD:

$$1.0 * (70-30)/(80-20) = .67$$

To disable prelayout scaling, set the `timing_prelayout_scaling` variable to `false`.

---

## Support for Fine-Grained Switch Cells in Power Analysis

PrimeTime supports the fine-grained switch cells that are defined in the library, using the Liberty syntax. A library cell is considered a fine-grained switch cell only when the cell is defined with the cell-level `switch_cell_type` attribute, as shown in the following example:

```
switch_cell_type: fine_grain;
```

Use the `report_power` command to view the power consumption of the fine-grained switch cells.

PrimeTime uses the voltage on the internal PG pins to determine the signal level of the pins that are related to the internal PG pins. Use the `set_voltage` command to specify design-specific voltage on the internal PG pins. If this is not specified, the voltage is derived from the `voltage_map` attribute in the library.

Note:

Using the `set_voltage` command on the external PG pins of the switch cell does not affect the voltage on its internal PG pins.

PrimeTime PX derives the `ON` and `OFF` states of the internal PG pins based on the `switch_function` and the `pg_function` definitions specified in the library for power analysis.

When you use the `connect_supply_net` command, PrimeTime supports explicit connection to the internal PG pins of fine-grained switch cells. Use the `report_power_pin_info` command to report the power on these pins.

Note:

The `report_power_pin_info` command reports the power of internal PG pins and corresponding external PG pins. The command does not report the power numbers separately for the internal and external PG pins.

For more information about defining a fine-grained switch, see the *Library Compiler Modeling Timing, Signal Integrity, and Power in Technology Libraries User Guide*. For more information about power analysis, see the Multivoltage Power Analysis chapter in the *PrimeTime PX User Guide*.

---

## Multivoltage Reporting and Checking

Several commands are available for reporting the power supply network and checking the network for errors or unusual conditions, including `get_*` commands, `report_*` commands, and `check_timing`.

---

### Collection (get\_\*) Commands

The `get_*` commands each create a collection of objects for reporting purposes. You can create a collection of existing power domains, power, switches, supply nets, or supply ports with the following commands.

```
get_power_domains
get_power_switches
get_supply_nets
```



```
get_supply_ports
get_supply_sets
```

The `get_power_domains` command returns a collection of power domains previously created with the `create_power_domain` command. You can pass the collection to another command for further processing, for example, to extract the name, scope, and elements attributes associated the domains. The `create_power_domain` command options allow you to limit the collection to the power domains meeting specified criteria. For example, the `-of_objects` option causes the collection to include only the power domains to which the specified objects belong. The `patterns` option limits the collection to power domains whose names match the specified pattern. The `get_power_switches`, `get_supply_nets`, and `get_supply_ports` commands operate in a similar manner and have similar command options.

[Table 11-1](#) lists the attributes of UPF power domain, supply net, supply port, and power switch objects.

*Table 11-1 UPF Collection Objects*

| Command and object class              | Attribute | Attribute type | Comment                                      |
|---------------------------------------|-----------|----------------|----------------------------------------------|
| get_power_domains<br>upf_power_domain | name      | string         | name as entered                              |
|                                       | full_name | string         | hierarchical name                            |
|                                       | elements  | collection     |                                              |
|                                       | scope     | string         |                                              |
|                                       | supplies  | string         | list of function keyword and supply set name |

Table 11-1 UPF Collection Objects (Continued)

| Command and object class            | Attribute   | Attribute type | Comment                                     |
|-------------------------------------|-------------|----------------|---------------------------------------------|
| get_supply_nets<br>upf_supply_net   | name        | string         | name as entered                             |
|                                     | full_name   | string         | hierarchical name                           |
|                                     | domains     | collection     |                                             |
|                                     | resolve     | string         | one_hot, etc.                               |
|                                     | voltage_min | float          | minimum-delay voltage                       |
|                                     | voltage_max | float          | maximum-delay voltage                       |
|                                     | static_prob | float          | probability of logic 1 (for power analysis) |
| get_supply_ports<br>upf_supply_port | name        | string         | name as entered                             |
|                                     | full_name   | string         | hierarchical name                           |
|                                     | domain      | collection     |                                             |
|                                     | direction   | string         | in, out                                     |
|                                     | scope       | string         |                                             |

*Table 11-1 UPF Collection Objects (Continued)*

| Command and object class               | Attribute               | Attribute type  | Comment                                                     |
|----------------------------------------|-------------------------|-----------------|-------------------------------------------------------------|
| get_supply_sets<br>upf_supply_set      | name                    | string          | name as entered                                             |
|                                        | full_name               | string          | hierarchical name                                           |
| get_power_switches<br>upf_power_switch | name                    | string          | name as entered                                             |
|                                        | full_name               | string          | hierarchical name                                           |
|                                        | domain                  | collection      |                                                             |
|                                        | output_supply_port_name | string          | name as entered                                             |
|                                        | output_supply_net       | collection      | supply net                                                  |
|                                        | input_supply_port_name  | string          | name as entered                                             |
|                                        | input_supply_net        | supply net      |                                                             |
|                                        | control_port_name       | list of strings | names as entered                                            |
|                                        | control_net             | collection      |                                                             |
|                                        | on_state                | list of strings | format: state_name<br>input_supply_port<br>boolean_function |

## Reporting Commands

These are the reporting commands associated with the power supply network:

```
report_power_domain
report_power_network
report_power_pin_info
report_power_switch
report_supply_net
report_supply_set
report_timing -voltage
```

The `report_power_domain` command reports the power domains previously defined with `create_power_domain` commands. The reports includes the names of the PG nets of each domain. For example,

```
pt_shell> report_power_domain [get_power_domains B]
...
Power Domain : B
Scope : H1
Elements : PD1_INST H2 H3

Connections : -- Power -- -- Ground --
Primary H1/H7/VDD H1/H7/VSS

```

The `report_power_network` command generates a report on all connectivity of the entire power network, through ports and switches. You can restrict the report to one or more specified nets. For example,

```
pt_shell> report_power_network -nets H1/VDD
...

Supply Net: H1/VDD
```

```
Connections:
Name Object type Domain

VDD Supply_port D1
H1/VDD Supply_port D2
U1/VDD PG_power_pin D1
SW1/IN Switch_input D1
```

The `report_power_pin_info` command reports the PG connectivity of leaf-level cells or library cells used in the design. For example,

```
pt_shell> report_power_pin_info [get_cells -hierarchical]
...
```

Note: Power connections marked by (\*) are exceptional

| Cell        | Power Pin Name | Type           | Voltage |        | Power Net Connected |
|-------------|----------------|----------------|---------|--------|---------------------|
|             |                |                | Max     | Min    |                     |
| PD0_INST/I0 | PWR            | primary_power  | 0.9300  | 1.2700 | int_VDD_5           |
| PD0_INST/I0 | GND            | primary_ground | 0.0000  | 0.0000 | A_VSS               |
| PD0_INST/I1 | PWR            | primary_power  | 0.9300  | 1.2700 | int_VDD_5           |
| PD0_INST/I1 | GND            | primary_ground | 0.0000  | 0.0000 | A_VSS               |
| I0          | PWR            | primary_power  | 1.0000  | 1.0000 | int_VDD_4 (*)       |
| I0          | GND            | primary_ground | 0.0000  | 0.0000 | int_VSS_4 (*)       |
| I1          | PWR            | primary_power  | 1.1500  | 1.1500 | T_VDD               |
| I1          | GND            | primary_ground | 0.0000  | 0.0000 | T_VSS               |
| PD1_INST/I0 | PWR            | primary_power  | 1.0000  | 1.0000 | int_VDD_4           |
| PD1_INST/I0 | GND            | primary_ground | 0.0000  | 0.0000 | int_VSS_4           |
| PD1_INST/I1 | PWR            | primary_power  | 1.0000  | 1.0000 | int_VDD_4           |

```
PD1_INST/I1 GND primary_ground 0.0000 0.0000 int_VSS_4
```

```
pt_shell> report_power_pin_info [get_lib_cells -of [get_cells -hier]]
```

```
...
```

| Cell   | Power Pin Name | Type           | Voltage |
|--------|----------------|----------------|---------|
| INVS2  | PWR            | primary_power  | 1.0000  |
| INVS2  | GND            | primary_ground | 0.0000  |
| BUFS2  | PWR            | primary_power  | 1.0000  |
| BUFS2  | GND            | primary_ground | 0.0000  |
| AND2X1 | PWR            | primary_power  | 1.0000  |
| AND2X1 | GND            | primary_ground | 0.0000  |
| OR2X1  | PWR            | primary_power  | 1.0000  |
| OR2X1  | GND            | primary_ground | 0.0000  |

The `report_power_switch` command reports the power switches in the design previously created with the `create_power_switch` command. Here is an example of a power switch report:

```
pt_shell> report_power_switch
```

```
...
```

```
Total of 3 power switches defined for design 'top'.
```

```
Power Switch : sw1
```

```

Power Domain : PD_SODIUM
Output Supply Port : vout VN3
Input Supply Port : vin1 VN1
Control Port: ctrl_small ON1
Control Port: ctrl_large ON2
Control Port: ss SUPPLY_SELECT
On State : full_s1 vin1 { ctrl_small & ctrl_large & ss }

```

```
...
```

The `report_supply_net` command reports the supply net information for a power domain or specified object in a power domain. For example,

```
pt_shell> report_supply_net
```

```
...
```

```
Total of 14 power nets defined.
```

```
Power Net 'VDD_Backup' (power)
```

```

Backup Power Hookups: A
```

```
Power Net 'VSS_Backup' (ground)
```

```

Backup Ground Hookups: A
```

```

Power Net 'T_VDD' (power,switchable)

Voltage states: {1.2}
Voltage ranges: {1.1 1.3}
Max-delay voltage: 1.15
Min-delay voltage: 1.15
Primary Power Hookups: T

Power Net 'A_VDD' (power)

Max-delay voltage: 1.15
Min-delay voltage: 1.15
Primary Power Hookups: A

```

The `report_supply_set` command reports detailed information about the supply sets defined in the design. For example,

```
pt_shell> report_supply_set
...
```

```
Total of 1 supply net defined for design "mydesign".
```

```

Supply Set : primary_sset
Scope : top
Function : power, supply net association: VDD
Funciton : ground, supply net association: VSS

```

The `report_timing -voltage` command allows you to determine the voltage of signal pins on timing paths. Similarly, you can use the `voltage` attribute on the `timing_point` objects in custom reports with the `get_timing_paths` command. The following example shows the `report_timing -voltage` command:

```
report_timing -voltage
...

```

| Point                            | Incr | Path   | Voltage |
|----------------------------------|------|--------|---------|
| clock HVCLK_SYNC_D (rise edge)   | 0.00 | 0.00   |         |
| clock network delay (propagated) | 0.00 | 0.00   |         |
| input external delay             | 5.00 | 5.00 f |         |
| in0[0] (in)                      | 0.00 | 5.00 f | 1.00    |
| ALU1/in0[0] (alu_ao_3)           | 0.00 | 5.00 f |         |
| ALU1/t0_reg_0/D (SDFQM1RA)       | 0.00 | 5.00 f | 1.03    |
| data arrival time                |      | 5.00   |         |

---

## Using the check\_timing Command

You can use the `check_timing` command to check the validity of the power supply connections in the design, including the following types of checks:

- Voltage set on each supply net segment (`supply_net_voltage`)
- Supply net connected to each PG pin of every cell (`unconnected_pg_pins`)
- Compatible signal levels between driver and load pins (`signal_level`)

### Voltage Set on Each Supply Net Segment

Every supply net segment must have a voltage set on it with the `set_voltage` command. To verify that this is the case, include a supply net voltage check in the `check_timing` command, as in the following example:

```
pt_shell> check_timing -include supply_net_voltage -verbose
Information: Checking 'no_clock'.
Information: Checking 'no_input_delay'.
...
Information: Checking 'supply_net_voltage'.
Warning: There are '2' supply nets without set_voltage. (UPF-029)
 VG
 VS1
Warning: The voltage of driving cell 'INV'(0.700) does not match the
related supply voltage at port 'reset'(1.080). (UPF-408)
Information: Checking 'pulse_clock_non_pulse_clock_merge'.
```

### Supply Net Connected to Each PG Pin of Every Cell

Each PG pin of every cell should be connected to a UPF supply net. To verify that this is the case, include an unconnected PG pin check in the `check_timing` command, as in the following example:

```
pt_shell> check_timing -include unconnected_pg_pins
...
```

Each UPF supply net connection can be either explicit (as specified by the `connect_supply_net` command) or implicit (due to the assignment of the cell to a power domain, isolation strategy, retention strategy, and so on).

### Compatible Driver-to-Load Signal Levels

To have PrimeTime check for mismatching voltage levels between cells that use different supply voltages, use the following `check_timing` command:

```
pt_shell> check_timing -include signal_level
```

This type of timing check traverses all nets and determines whether the output voltage level of each driver is sufficient to drive the load pins on the net. PrimeTime reports any driver-load pairs that fail the voltage level check. It assumes that there is no voltage degradation along the net. You can fix a violation by changing the supply voltages or by inserting a level shifter between the driver and load.

PrimeTime performs signal level checking by comparing the input and output voltage levels defined in the library for the pins of leaf-level cells. The checked signal levels are based on either the gate noise immunity (or signal level range) margins defined by the Liberty syntax `input_voltage` and `output_voltage` or a comparison of driver/load supply voltages. You can control driver and load supply voltage mismatch reporting by using the `set_level_shifter_strategy` and `set_level_shifter_threshold` commands.

The `set_level_shifter_strategy` command specifies the type of strategy used for reporting voltage mismatches: `all`, `low_to_high`, or `high_to_low`. The `low_to_high` strategy reports the voltage level mismatches when a source at a lower voltage drives a sink at a higher voltage. The `high_to_low` strategy reports the voltage level mismatches when a source at a higher voltage drives a sink at a lower voltage. The `all` strategy (the default) reports both types of mismatches.

The `set_level_shifter_threshold` command specifies the absolute and relative mismatch amounts that trigger an error condition. If there is a voltage difference between driver and load, the difference must be less than both the absolute and relative thresholds to be considered acceptable. For example, the following command sets the voltage difference threshold to 0.1 volt and the percentage threshold to 5 percent:

```
pt_shell> set_level_shifter_threshold -voltage 0.1 -percent 5
```

A voltage difference that is more than 0.1 volt or more than five percent of the driver voltage is reported as a mismatch. The default thresholds are both zero, so if you set one threshold to a nonzero value, you should set the other to a nonzero value as well. To disable either absolute or percentage threshold checking, set its threshold to a large value.

The percentage difference is determined as follows:

$$\text{abs}(\text{driver}(\text{VDD}) - \text{load}(\text{VDD})) / \text{driver}(\text{VDD}) * 100$$

PrimeTime reports either of the following conditions as an “incompatible voltage” error:

- Driver  $V_{Omax} > \text{Load } V_{Imax}$
- Driver  $V_{Omin} < \text{Load } V_{Imin}$

PrimeTime reports either of the following conditions as a “mismatching driver-load voltage” warning:

- Driver  $V_{OH} < \text{Load } V_{IH}$



- Driver VOL > Load VIL

Here is an example of a voltage mismatch report:

```
pt_shell> check_timing -verbose -include signal_level
```

```
...
Error: There are 2 voltage mismatches
MIN-MAX - driver vomax > load vimax:
```

| Driver | Voltage | Load | Voltage | Margin |
|--------|---------|------|---------|--------|
| u2/Z   | 2.50    | u3/A | 0.90    | -1.60  |
| u3/Z   | 2.10    | u5/A | 1.47    | -0.63  |

```
Warning: There is 1 voltage mismatch
MIN-MAX - driver vol > load vil:
```

| Driver | Voltage | Load | Voltage | Margin |
|--------|---------|------|---------|--------|
| u2/Z   | 0.30    | u3/A | 0.20    | -0.10  |

The technology library defines the input and output voltages in terms of the supply voltage in Liberty syntax. PrimeTime correctly calculates the voltages for comparison. For example, a library might define the input and output voltage levels as follows:

- $VIL = 0.3 * VDD$ ,  $VIH = 0.7 * VDD$
- $VOL = 0.4$ ,  $VOH = 2.4$
- $Vlmin = -0.5$ ,  $Vlmax = VDD + 0.5$
- $VOmin = -0.3$ ,  $VOmax = VDD + 0.3$

PrimeTime calculates the input and output voltages, taking into account the supply voltages that apply to each cell.

For proper transition time scaling for cells with multiple power rails, PrimeTime requires that you define the `input_signal_level` and `output_signal_level` attributes on multirail cells. For `check_timing -include signal_level`, PrimeTime uses the definitions of the `input_voltage` and `output_voltage` attributes on all pins.

Even without `input_voltage` or `output_voltage` specified, PrimeTime still attempts to report the 100 worst mismatches based on rail voltages not being exactly equal. For example,

```
pt_shell> check_timing -include signal_level -verbose
```

```
...
Warning: There are 2 voltage mismatches
MAX-MAX - driver rail != load rail:
The 100 worst voltage mismatches:
```

| Driver | Voltage | Load    | Voltage | Margin |
|--------|---------|---------|---------|--------|
| u1/Y   | 4.75    | u3/A    | 3.00    | -1.75  |
| u3/Y   | 3.00    | ff2/CLK | 4.75    | -1.75  |

Warning: There are 2 voltage mismatches  
 MIN-MIN - driver rail != load rail:  
 The 100 worst voltage mismatches:

| Driver | Voltage | Load    | Voltage | Margin |
|--------|---------|---------|---------|--------|
| u1/Y   | 5.25    | u3/A    | 3.00    | -2.25  |
| u3/Y   | 3.00    | ff2/CLK | 5.25    | -2.25  |
| ...    |         |         |         |        |

The following table summarizes the effects of the threshold and strategy settings.

| Signal level<br>check type                        | Strategy impact |                                                                    |                                                                    | Threshold<br>impact                                    | Explanation                                                                                                            |
|---------------------------------------------------|-----------------|--------------------------------------------------------------------|--------------------------------------------------------------------|--------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------|
|                                                   | All             | Low_to_high                                                        | High_to_low                                                        |                                                        |                                                                                                                        |
| Driver V <sub>Omax</sub> > Load V <sub>lmax</sub> | None            | None                                                               | None                                                               | None                                                   | Driver exceeding breakage voltage of the load cell defined by the library is a fatal condition. It cannot be filtered. |
| Driver V <sub>Omin</sub> < Load V <sub>lmin</sub> | None            | None                                                               | None                                                               | None                                                   |                                                                                                                        |
| Driver V <sub>OH</sub> < Load V <sub>IH</sub>     | None            | None                                                               | None                                                               | None                                                   |                                                                                                                        |
| Driver V <sub>OL</sub> > Load V <sub>IL</sub>     | None            | None                                                               | None                                                               | None                                                   | Library-defined signal level margins define safe region of operation and cannot be suppressed.                         |
| Driver VDD != Load VDD                            | None            | Filter any mismatch<br>VDD <sub>driver</sub> > VDD <sub>load</sub> | Filter any mismatch<br>VDD <sub>driver</sub> > VDD <sub>load</sub> | Further filter any mismatch smaller than the tolerance | The user-defined tolerance and strategy are fully used.                                                                |

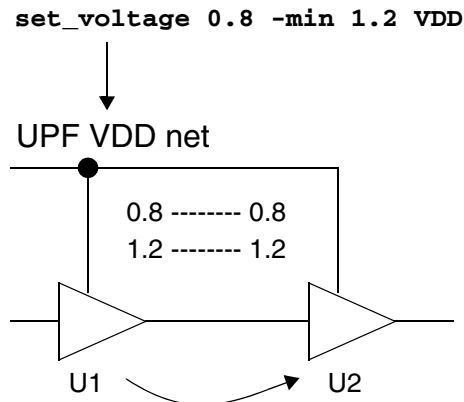
## Impact of Correlated Supplies on Signal Level Checking

The signal level check correctly considers the case where minimum and maximum voltages are set on the power nets that supply both of the cells being compared. For example, suppose that you set the voltage for supply VDD as follows:

```
pt_shell> set_voltage 0.8 -min 1.2 VDD # corner voltages
```

This command sets the supply voltage to 0.8 for maximum-delay analysis and 1.2 Volts for minimum-delay analysis. Suppose that the same VDD supply applies to both the driver and receiver cells of a net, as shown in [Figure 11-1](#).

*Figure 11-1 Signal Level Checking With Supply Correlation*



PrimeTime considers the fact that the supply voltages of the driver and receiver are correlated. When the supply voltage is low for U1, it is also low for U2, so there is no mismatch. The same is true for the case where the supply voltage is high.

## UPF Supply Sets

A supply set is a higher-level abstraction of the supply net enabling you to defer specifying supply nets and supply ports to later in the design stage. A supply set relates multiple supply nets as a complete power source for one or more design elements. Each supply net in a supply set provides a function that can be used just as any other supply net in the UPF design, with no domain restrictions. When a supply set is updated with actual supply nets, the supply nets must also be domain independent. A domain independent supply net is a supply net created using the `create_supply_net` command, but without specifying the `-domain` option. After creation, the supply net is available to be used in any power domain at the current scope and below.

The supply set is domain independent, and it supports the power and ground supply net functions to be used in the design. You can define a supply set that specifies the power network in the current logic hierarchy by using the `create_supply_set` UPF command. This command creates a supply set in the current scope. A supply set eliminates the requirement of a supply net and supply ports in the design in the front-end tool. However, before physical synthesis occurs, actual supply nets must be associated with the supply sets.

The supply net can be referenced by either the actual supply net or the supply net handle, such as `supply_set_name.power` and `supply_set_name.ground`. If you use the `-update` option of the `create_supply_set` command, PrimeTime associates an actual supply net to the power and ground functional capabilities of an existing supply set. Just like the supply set, the actual supply net must be domain independent. The actual supply net can be referenced through the supply set or by the actual net. The following UPF commands support supply net handles:

```
connect_supply_net
create_power_switch
set_domain_supply_net
set_design_attributes
set_port_attributes
set_isolation
set_retention
set_related_supply_net
set_voltage
```

You can associate supply sets with the corresponding supply set handles of a power domain by using the `create_power_domain` command. You can also associate supply sets with isolation or retention supply nets, such as power or ground nets, of a specific isolation strategy using the `set_isolation` command or a specific retention strategy by using the `set_retention` command. For more information, see the `create_power_domain`, `set_isolation`, and `set_retention` man pages.

---

## Library PG Tcl File

PG library conversion and update information is specified in a PG Tcl file. You can generate this file with Design Compiler or IC Compiler. Otherwise, you can manually create the file. You can set the `library_pg_file_pattern` variable to specify the location of the PG Tcl file for library PG conversion and update feature. By setting the variable to a valid file name pattern, the tool locates the associated PG Tcl file for the technology libraries that are provided.

You can read the technology library files into PrimeTime using the `read_lib` command or into PrimeTime PX using the external reader (ptxr). You can also use the updated database written out by Library Compiler. At the time the libraries are loaded, the tool performs automatic library PG updates on the in-memory databases.

By default, the `library_pg_file_pattern` variable is set to "" (empty string) so that there is no PG Tcl side file. You can set this variable to use one Tcl file for all databases, one per group of databases, or one per database. You set the variable before loading the library by specifying the following name pattern, where *string* is either the path to the database directory or the leaf file name for the database:

```
pt_shell> set library_pg_file_pattern string
```

To specify one Tcl file for all databases, you would use a command similar to the following example:

```
pt_shell> set library_pg_file_pattern "libpg_sidefile.pg"
```

To specify one Tcl file per database at the same location as the original database files, you would use a command similar to the following example:

```
pt_shell> set library_pg_file_pattern "__DIR__/__FILE__.pg"
```

---

## Default Power and Ground Pin Names

You can use a non-PG library with UPF-specified power intent. When you load a non-PG library into memory without specifying a PG Tcl side file, PrimeTime can create one default power pin and one default ground pin for each cell.

You specify the names assigned to the power and ground pins by setting the `lp_default_power_pin_name` and `lp_default_ground_pin_name` variables. These variables are initially set to an empty string, so you must set them explicitly to the desired power and ground pin names before you load the non-PG library. For example,

```
pt_shell> set lp_default_power_pin_name "VDD"
pt_shell> set lp_default_ground_pin_name "VSS"
```

The specified names are used for naming the newly created power and ground pins of the library cells stored in memory and for creating the voltage map defined at the library level. Only library cells from non-PG libraries are affected by these variable settings, and only when there is no PG Tcl side file specified by the `library_pg_file_pattern` variable.

When library loading occurs within PrimeTime, you receive messages about the PG pins being added to the library. If your flow requires multiple loads of the library, you might see these messages more than once. For more information about the default PG pin naming variables, see the specific man page.

---

## Power Domain Mode of Release Z-2007.06

If you want to use the power domain flow based on setting object-specific voltages and operating conditions, set the `power_domains_compatibility` variable to `true`. This causes PrimeTime to enter the “power domain mode,” which enables the Z-2007.06 multivoltage syntax, disables the UPF syntax, and removes any UPF-specified power supply data in the design. Due to differences in command syntax and data infrastructure, the Z-2007.06 power domain capabilities are not compatible with the UPF features.

When PrimeTime is in the power domain mode, you can view the list of power domain commands with the following command:

```
pt_shell> help "power domains"
Power Domains:
connect_power_domain # Connect power domain
connect_power_net_info # Connect power net to cell power pin
create_power_domain # Create power domain object
create_power_net_info # Create power net info object
get_power_domains # Create a collection of power domains
report_power_domain # Report power domain
report_power_net_info # Report all power net objects
report_power_pin_info # Report power pin info
set_voltage # Set voltage
```

There is some overlap in command syntax between the power domain mode and the UPF mode. The interpretation of a command such as `create_power_domain` depends on the operating mode.

Physical tools such as IC Compiler and Design Compiler create power distribution networks based on user-specified connectivity in Tcl syntax. PrimeTime SI can use the same set of commands to build a virtual model of the power distribution network. The power domain model, together with the `set_voltage` command, allows PrimeTime SI to determine the voltage at the power pins of each leaf-level instance. These are the power domain commands:

- The `create_power_net_info` command specifies the name of a power supply net or ground net in the design. If it is a power supply, the command also specifies the voltage and whether the power can be switched off.
- The `create_power_domain` command specifies the name of a power domain and lists the hierarchical cells associated with the domain. If no list is provided, the power domain applies to the top level; there can be no more than one such top-level domain. The command also specifies whether the domain can be powered down and if so, the control net.

- The `connect_power_domain` command creates logical power connections for a specified power domain. It specifies the primary, backup, and internal PG nets for a specified power domain. All cells in the power domain inherit the specified power connections.

The backup PG nets are for always-on logic, retention registers, isolation cells, and enable level shifter cells. The internal PG nets are for the switching cells inside the power domain.

- The `connect_power_net_info` command makes power net connections for a specific power pin of a leaf cell. The pin-level connections override the domain-level connections made with the `connect_power_domain` command.
- The `set_voltage` command defines the operating voltage on the power nets defined by the `create_power_nets_info` command. You can specify a single voltage or a minimum and a maximum voltage for the power net. If you do not use this command, the available operating condition settings are used.

If you link the design again, the power domain information is discarded.

A sequence of commands similar to the following can be used to perform timing analysis of a multivoltage design:

```
Enable power domains (non-UPF) mode
set power_domains_compatibility TRUE
Read libraries, designs
...
read_lib l1.db
read_verilog d1.v
...

Read libraries, design, SDC, and
update design and setup link to HSPICE

create_power_net_info -power power_net_name

create_power_domain domain_name -object_list object_list

connect_power_domain -primary_power_net power_net \
 -backup_power_net power_net

connect_power_net_info object_list \
 -power_pin_name pin_name -power_net_name net_name

Read SDC and other timing assertions
source d1.tcl

Read SDC and other timing assertions
set_voltage on power nets or IR drop on power pins

Perform timing, signal integrity analysis
report_timing
```

The following commands report information about the power rails.

- The `report_power_domain` command reports the power domains in the design and their connections. If you specify a list of objects, only the power domains of those objects are reported.
- The `report_power_net_info` command reports the names of the power nets and their associated power domain hookups.
- The `report_power_pin_info` command reports power pin information for library cells or for leaf-level cells in the current design. Specify a list of library cells to find out the names, types, and voltage specifications of the power pins in the library cells. Specify a list of leaf-level cells in the design to find out the power net connections to the power pins of those cells.

---

## Multivoltage Method Prior to Release Z-2007.06

To perform multivoltage analysis, you can specify multiple voltages by applying different operating conditions to different cells with the `set_operating_conditions -object_list` command, by directly annotating voltage drop information with the `set_voltage` command, or by linking blocks to different libraries. (Using the `set_voltage` command requires a PrimeTime SI license.) Any subsequent timing analysis takes into account the supply voltages specified for the design. The `report_cell` command reports the instance-specific operating conditions and supply voltage information.

These commands were available prior to the Z-2007.06 release and are still supported. However, if you use any UPF commands in a session, the pre-Z-2007.06 multivoltage specification method is disabled, and only UPF commands can be used for specifying power supply information. In that case, you can no longer use `set_operating_conditions -object_list` or `set_voltage`, and any multivoltage information previously set with those commands is discarded.

In the pre-Z-2007.06 multivoltage flow, there are two ways to set supply voltages on a linked design:

- Create different operating conditions having different rail voltages, either by defining them in the technology library or by using the `create_operating_conditions` command. Then apply different operating conditions to different hierarchical blocks or leaf-level cells in the design with the `set_operating_conditions -object_list` command. This method is appropriate for specifying voltage islands on the chip.
- Use the `set_voltage` command to set rail voltages directly on hierarchical blocks or leaf-level cells, overriding any applicable operating condition voltages. This method is appropriate for back-annotating voltage drop information from a power rail analysis tool.



You can combine these two methods in the same design. To do so, first apply the operating conditions to define the voltage islands, then apply the voltage drop information. Rail voltages set with the `set_voltage` command override those set with operating conditions on a cell-by-cell basis. However, they do not override operating conditions set at lower levels of hierarchy.

If you do not set conditions with `set_operating_conditions`, PrimeTime SI uses the default operating conditions of the individual libraries that the cells came from. To have the cells use the operating conditions from the main library instead (the first library defined in the link path), set the `default_oc_per_lib` variable to `false`.

To specify different NLDM library cells for different instances in the design, set the `link_path_per_instance` variable to a list, with each list element consisting of a list of instances and the corresponding link paths that override the default link path for each of those instances. For an example, see [“Analysis With Multiple Voltages” on page 11-10](#).

---

## Setting Operating Conditions on Cells

By default, the `set_operating_conditions` command sets the operating conditions for the whole current design. To override global operating conditions and set the conditions for a hierarchical block or cell instance, use the `-object_list` option and list the applicable cells. For example,

```
pt_shell> set_operating_conditions WCCOM5.0
pt_shell> set_operating_conditions \
 -object_list ALU/A4 WCCOM3.5
```

The first command specifies the set of operating conditions for the whole chip. The second command overrides the whole-chip settings for the cell ALU/A4.

You can also set operating conditions on ports. The specified operating conditions apply to the driving cells for the ports (as specified by the `set_driving_cell` command).

With different operating conditions applied to different levels of the cell hierarchy, the lowest-level setting for a cell has priority over higher-level settings. To get a report on the applicable minimum and maximum operating condition settings for a cell, use the `report_cell` command.

---

## Setting Rail Voltages Directly on Cells

To set the rail voltage on a hierarchical block or leaf-level cell (irrespective of operating conditions), use the `set_rail_voltage` command. Using this command requires a PrimeTime SI license. The command is used in a manner similar to the `set_operating_conditions -object_list` command. For example,

```
pt_shell> set_rail_voltage -max -rail_value 1.2 [get_cells {ALU/a5}]
```

```
pt_shell> set_rail_voltage -min -rail_value 1.4 [get_cells {ALU/a5}]
```

# 12

## Object Attributes

---

An attribute is a string or value associated with an object in the design that carries some information about that object. For example, the `number_of_pins` attribute attached to a cell indicates the number of pins in the cell. You can write programs in Tcl to get attribute information from the design database and generate custom reports on the design.

The following sections describe attributes and how to use them:

- [Using Attributes](#)
- [Saving Design Attributes](#)
- [Attribute Names and Usage](#)
- [Using Paths to Generate Custom Reports](#)
- [Using Arcs to Generate Custom Reports](#)

---

## Using Attributes

PrimeTime provides a set of commands for setting, reporting, listing, and creating attributes, as summarized in [Table 12-1](#).

*Table 12-1 Attribute Commands*

| Attributes                         | Description                                                                                                              |
|------------------------------------|--------------------------------------------------------------------------------------------------------------------------|
| <code>list_attributes</code>       | Shows the attributes defined for each object class or a specified object class; optionally shows application attributes. |
| <code>get_attribute</code>         | Retrieves the value of any attribute from a single object.                                                               |
| <code>report_attribute</code>      | Displays the value of all attributes on one or more objects; optionally shows application attributes.                    |
| <code>define_user_attribute</code> | Creates a new attribute for one or more classes.                                                                         |
| <code>set_user_attribute</code>    | Sets a user-defined attribute on one or more objects.                                                                    |
| <code>remove_user_attribute</code> | Removes a user-defined attribute from one or more objects.                                                               |

---

The `get_attribute`, `report_attribute`, `set_user_attribute`, and `remove_user_attribute` commands accept an object specification, which can be a collection or a list of collections. The object specification for the `get_attribute` command is limited to one collection containing one object.

---

## Defining User Attributes

The `define_user_attribute` command defines a new attribute. PrimeTime has a set of attributes that it considers application attributes. You can view these attributes by using `list_attributes -application`. You must define any other attribute before you can use it in PrimeTime. One such use is importing an attribute from the `.ddc` or `.db` file.

You can apply attributes to most object classes in PrimeTime. These can mark interesting cells or nets, store values you have computed, and so on. PrimeTime cannot use these attributes, but you can use them in scripts, procedures, and so on. You can list the attributes you define using the `list_attributes` command. When you define an attribute, decide on the appropriate data type. For more information, see the `define_user_attribute` man page.

Use the following commands to define attributes of various types. You can specify more than one class. Enter

```
pt_shell> define_user_attribute attr_s -class {cell net} -type string
pt_shell> define_user_attribute attr_i -class cell -type int
```

The following command sequence defines attribute `attr_irl` as  $\geq 0$  and  $\leq 100$ , attribute `attr_ir2` as  $\geq 0$  with no maximum, and attribute `attr_ir3` as  $\leq 100$  with no minimum.

```
pt_shell> define_user_attribute attr_irl -class cell -type int \
 -range_min 0 -range_max 100
pt_shell> define_user_attribute attr_ir2 -class cell -type int -range_min
0
pt_shell> define_user_attribute attr_ir3 -class cell -type int -range_max
100
```

The following command defines attribute `attr_oo` for cells. The attribute is a string, but you can set it only to A, B, C, or D. Enter

```
pt_shell> define_user_attribute attr_oo \
 -class cell -type string -one_of {A B C D}
```

---

## Importing User-Defined Attributes

To load user-defined attributes (attributes that are not PrimeTime application attributes) from a .ddc or .db file into PrimeTime, you must define the attributes in PrimeTime before you read any designs.

Assume you created a Boolean attribute in Design Compiler called `MarkedPin` and placed this attribute on several pins. In PrimeTime, before you read any designs you must define the `MarkedPin` attribute. For example, enter

```
pt_shell> define_user_attribute -class pin -type Boolean -import
MarkedPin
```

After you define the attribute in this way, PrimeTime extracts it from the .ddc or .db file as it would extract any other attribute. For example, enter

```
pt_shell> list_attributes -class pin

Report : List of Attribute Definitions
Design :
Version: 2000.11
Date : Tues Nov 12 13:25:04 2000

```

**Properties:**

- A - Application-defined
- U - User-defined
- I - Importable from db (for user-defined)

| Attribute Name | Object | Type    | Properties | Constraints |
|----------------|--------|---------|------------|-------------|
| -----          | -----  | -----   | -----      | -----       |
| MarkedPin      | pin    | Boolean | U,I        |             |

Attributes appear on design objects only after the design is linked.

User-defined attributes can be imported from both top-level designs and lower-level designs. To import user-defined attributes at the top level, there are no special considerations. You can use the `define_user_attribute -import` command; however, note that a top-level design attribute is attached to a design, but an inherited (lower-level) design attribute is attached to an instance of that design, which is a cell. Therefore, to import and inherit a lower-level design attribute, the attribute must be defined for both designs and cells. The `-import` option is only needed at the design level.

Similarly, a top-level port attribute is attached to a port, but an inherited port attribute is attached to an instance of that port, which is a pin. Therefore, to import and inherit a port attribute, the attribute must be defined for both ports and pins. The `-import` option is only needed at the port level.

To import and inherit a design attribute, the attribute must be defined for both designs and cells, although it only needs to be imported for the design. To import and inherit a port attribute, the attribute must be defined for both ports and pins, although it only needs to be imported for the port. A required attribute is defined automatically if necessary to import a related attribute. After the design is linked, you can see that user-defined attributes have been applied. For example,

```
pt_shell> report_attribute [get_pins */CP]
```

```

Report : Attribute
Design : M
Version: 2000.11
Date : Tues Nov 14 13:33:25 2000

```

| Design | Object    | Type    | Attribute Name | Value |
|--------|-----------|---------|----------------|-------|
| -----  | -----     | -----   | -----          | ----- |
| M      | o_reg1/CP | Boolean | MarkedPin      | true  |
| M      | o_reg2/CP | Boolean | MarkedPin      | true  |
| M      | o_reg3/CP | Boolean | MarkedPin      | true  |
| M      | o_reg4/CP | Boolean | MarkedPin      | true  |

You can create a collection of all pins with this attribute. Enter

```
pt_shell> set ipins [get_pins * -hier -filter \
 "MarkedPin == true"]
```

---

## Saving Design Attributes

By default, PrimeTime saves nothing when it exits. To save design attributes you applied during a session, use the `write_script` command.

The `write_script` command writes the following information:

- Clock-related information: Clock creation, generated clock creation, clock latency, clock uncertainty, and interclock uncertainty
- Timing-related information: Disable timing, maximum time borrow
- Point-to-point exceptions: false path, minimum delay, maximum delay, multicycle path, group path, input delay, output delay, annotated delay, and annotated checks
- Net attributes: Capacitance and resistance
- Port attributes: Fanout, capacitance, and resistance
- Design environment: Wire load model, operating condition, drive, driving cell, and input transition
- Design rules: minimum capacitance, maximum capacitance, minimum transition, maximum transition, minimum fanout, and maximum fanout

You can save the design attributes in Synopsys `pt_shell`, `dc_shell`, or `dctcl` script formats, then use the script to re-create the attributes on the design. For more information, see the `write_script` man page.

---

## Attribute Names and Usage

The PrimeTime `get_attribute` command supports the attributes listed in the tables in this appendix. PrimeTime attributes are read-only (you cannot set them) unless otherwise specified. For information about attributes used in PrimeTime SI, see the *PrimeTime SI User Guide*. Find the attributes by class in the following sections:

- [Cell Object Class Attributes](#)
- [Clock Object Class Attributes](#)
- [Design Object Class Attributes](#)

- [Library Object Class Attributes](#)
- [Library Cell Object Class Attributes](#)
- [Library Pin Object Class Attributes](#)
- [Library Timing Arc Object Class Attributes](#)
- [Net Object Class Attributes](#)
- [Path Group Object Class Attributes](#)
- [Pin Object Class Attributes](#)
- [Port Object Class Attributes](#)
- [Timing Arc Object Class Attributes](#)
- [Timing Path Object Class Attributes](#)
- [Timing Point Object Class Attributes](#)

---

## Cell Object Class Attributes

[Table 12-2](#) lists the cell object class attributes.

*Table 12-2 Attributes of the cell Object Class*

| Attribute name  | Type    | Description                                                                                                                                                                                      |
|-----------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| area            | float   | The area of a cell. If the cell is hierarchical, this includes net area.                                                                                                                         |
| base_name       | string  | The leaf name of a cell. For example, the <code>base_name</code> of cell U1/U2/U3 is U3.                                                                                                         |
| bottleneck_cost | float   | The bottleneck cost computed by the <code>report_bottleneck</code> command requires running this command before querying the attribute.                                                          |
| disable_timing  | Boolean | This attribute is <code>true</code> if the timing for the cell has been marked as disabled in <code>set_disable_timing</code> . You can set and unset the <code>disable_timing</code> attribute. |



*Table 12-2 Attributes of the cell Object Class (Continued)*

| Attribute name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dont_touch                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | Boolean | Identifies cells to be excluded from optimization in Design Compiler. Values are undefined by default. Cells with the dont_touch attribute set to true are not modified or replaced during compilation in Design Compiler. Set with the set_dont_touch command and used by characterize_context and create_timing_context. For more information, see the set_dont_touch Design Compiler man page. |
| dynamic_power                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | double  | Shows the dynamic power of a cell in watts. It is the sum of the internal power and switching power.                                                                                                                                                                                                                                                                                              |
| early_cell_check_derate_factor<br>early_clk_cell_derate_factor<br>early_clk_net_delta_derate_factor<br>early_clk_net_derate_factor<br>early_data_cell_derate_factor<br>early_data_net_delta_derate_factor<br>early_data_net_derate_factor<br>early_fall_cell_check_derate_factor<br>early_fall_clk_cell_derate_factor<br>early_fall_clk_net_delta_derate_factor<br>early_fall_clk_net_derate_factor<br>early_fall_data_cell_derate_factor<br>early_fall_data_net_delta_derate_factor<br>early_fall_data_net_derate_factor<br>early_rise_cell_check_derate_factor<br>early_rise_clk_cell_derate_factor<br>early_rise_clk_net_delta_derate_factor<br>early_rise_clk_net_derate_factor<br>early_rise_data_cell_derate_factor<br>early_rise_data_net_delta_derate_factor<br>early_rise_data_net_derate_factor | float   | Early timing derate factors, specified using the set_timing_derate command, that apply to the cell. The net derates listed here are only defined for hierarchical cells.                                                                                                                                                                                                                          |
| escaped_full_name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | string  | Contains the name of the cell. Any literal hierarchy characters are escaped with a backslash.                                                                                                                                                                                                                                                                                                     |

*Table 12-2 Attributes of the cell Object Class (Continued)*

| Attribute name                              | Type                 | Description                                                                                                                                                                                        |
|---------------------------------------------|----------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>full_name</code>                      | <code>string</code>  | The complete name of a cell. For example, the full name cell U3 within cell U2 within cell U1 is U1/U2/U3. The <code>full_name</code> attribute is not affected by <code>current_instance</code> . |
| <code>gate_leakage_power</code>             | <code>double</code>  | The gate leakage power of a cell in watts. Gate leakage power is the leakage power from the source to the gate or the gate to the drain.                                                           |
| <code>glitch_power</code>                   | <code>double</code>  | The glitch power of a cell in watts. Glitch power is considered part of the dynamic power.                                                                                                         |
| <code>has_multi_ground_rails</code>         | <code>Boolean</code> | The attribute is <code>true</code> if a cell has multiple ground rails.                                                                                                                            |
| <code>has_multi_power_rails</code>          | <code>Boolean</code> | The attribute is <code>true</code> if a cell has multiple power rails.                                                                                                                             |
| <code>has_rail_specific_power_tables</code> | <code>Boolean</code> | The attribute is <code>true</code> if a cell has power tables attached to rails.                                                                                                                   |
| <code>internal_power</code>                 | <code>double</code>  | The internal power of a cell in watts. Internal power is any dynamic power dissipated within the boundary of a cell.                                                                               |
| <code>internal_power_derate_factor</code>   | <code>float</code>   | Power derate factors, specified using the <code>set_power_derate</code> command, that apply to the cell.                                                                                           |
| <code>intrinsic_leakage_power</code>        | <code>double</code>  | The intrinsic leakage power of a cell in watts. Most intrinsic leakage power results from source-to-drain subthreshold leakage.                                                                    |
| <code>is_black_box</code>                   | <code>Boolean</code> | This attribute is <code>true</code> if the cell's reference is not linked to a library cell or design. This attribute is read-only; you cannot change the setting.                                 |

*Table 12-2 Attributes of the cell Object Class (Continued)*

| Attribute name                               | Type    | Description                                                                                                                                                                                                        |
|----------------------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>is_clock_gating_check</code>           | Boolean | The attribute is <code>true</code> if a cell is a clock-gating check cell.                                                                                                                                         |
| <code>is_clock_network_cell</code>           | Boolean | The attribute is <code>true</code> if the cell is in the clock network of any clock in the design.                                                                                                                 |
| <code>is_combinational</code>                | Boolean | A cell is combinational if it is nonsequential or non-three-state and all of its outputs compute a combinational logic function. The <code>report_lib</code> command reports such a cell as not being a black box. |
| <code>is_design_mismatch</code>              | Boolean | This attribute is <code>true</code> if an object is dirty, that is, the object is directly impacted by a mismatch between the block and top-level design.                                                          |
| <code>is_edited</code>                       | Boolean | This attribute is <code>true</code> if the hierarchical cell has been uniquified as a result of a netlist editing (ECO) change. It is only defined for hierarchical cells.                                         |
| <code>is_fall_edge_triggered</code>          | Boolean | This attribute is <code>true</code> if a cell is used in the design as a falling-edge-triggered flip-flop.                                                                                                         |
| <code>is_hierarchical</code>                 | Boolean | This attribute is <code>true</code> for any pins of a hierarchical cell. It is <code>false</code> for pins of library cells, also known as leaf cells.                                                             |
| <code>is_ideal</code>                        | Boolean | This attribute is <code>true</code> if the cell has been marked ideal using the <code>set_ideal_network</code> command.                                                                                            |
| <code>is_integrated_clock_gating_cell</code> | Boolean | This attribute is <code>true</code> if a cell is defined in the library as an integrated clock-gating cell.                                                                                                        |

*Table 12-2 Attributes of the cell Object Class (Continued)*

| Attribute name                           | Type    | Description                                                                                                                                                                                                                                        |
|------------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>is_interface_logic_model</code>    | Boolean | This attribute is <code>true</code> if a cell is an interface logic model (ILM) of a design. You can set this attribute.                                                                                                                           |
| <code>is_macro_switch</code>             | Boolean | This attribute is <code>true</code> if the reference library cell is defined as a fine-grain switch cell in the library.                                                                                                                           |
| <code>is_memory_cell</code>              | Boolean | This attribute is inherited from the reference library cell. It is used by PrimeTime PX to identify memory cells when reporting power or activity annotation for different cell types, such as sequential, combinational, and memories cell types. |
| <code>is_mux</code>                      | Boolean | This attribute is <code>true</code> if a cell is a multiplexer.                                                                                                                                                                                    |
| <code>is_negative_level_sensitive</code> | Boolean | This attribute is <code>true</code> if a cell is used in the design as a negative level-sensitive latch.                                                                                                                                           |
| <code>is_pad_cell</code>                 | Boolean | This attribute is <code>true</code> if the cell is a pad cell.                                                                                                                                                                                     |
| <code>is_positive_level_sensitive</code> | Boolean | This attribute is <code>true</code> if a cell is used in the design as a positive level-sensitive latch.                                                                                                                                           |
| <code>is_power_standby_cell</code>       | Boolean | This attribute is <code>true</code> if a cell is a power standby cell.                                                                                                                                                                             |
| <code>is_rise_edge_triggered</code>      | Boolean | This attribute is <code>true</code> if a cell is used in the design as a rising-edge-triggered flip-flop.                                                                                                                                          |
| <code>is_sequential</code>               | Boolean | A cell is sequential if it is not combinational.                                                                                                                                                                                                   |
| <code>is_three_state</code>              | Boolean | This attribute is <code>true</code> if a cell is a three-state device.                                                                                                                                                                             |

*Table 12-2 Attributes of the cell Object Class (Continued)*

| Attribute name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | Type    | Description                                                                                                                                                                                                                                      |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| late_cell_check_derate_factor<br>late_clk_cell_derate_factor<br>late_clk_net_delta_derate_factor<br>late_clk_net_derate_factor<br>late_data_cell_derate_factor<br>late_data_net_delta_derate_factor<br>late_data_net_derate_factor<br>late_fall_cell_check_derate_factor<br>late_fall_clk_cell_derate_factor<br>late_fall_clk_net_delta_derate_factor<br>late_fall_clk_net_derate_factor<br>late_fall_data_cell_derate_factor<br>late_fall_data_net_delta_derate_factor<br>late_fall_data_net_derate_factor<br>late_rise_cell_check_derate_factor<br>late_rise_clk_cell_derate_factor<br>late_rise_clk_net_delta_derate_factor<br>late_rise_clk_net_derate_factor<br>late_rise_data_cell_derate_factor<br>late_rise_data_net_delta_derate_factor<br>late_rise_data_net_derate_factor | float   | Late timing derate factors, specified using the <code>set_timing_derate</code> command, that apply to the cell. The net derates listed here are only defined for hierarchical cells.                                                             |
| leakage_power                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | double  | The leakage power of a cell in watts. It is the power that a cell dissipates when it is not switching. Leakage power is the sum of the intrinsic leakage and gate leakage.                                                                       |
| leakage_power_derate_factor                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | float   | Power derate factors, specified using the <code>set_power_derate</code> command, that apply to the cell.                                                                                                                                         |
| number_of_pins                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | integer | Number of pins on the cell. The number of pins can be different before and after linking. For example, if some pins were unconnected in a Verilog instance, after linking to the lower-level design, additional pins can be created on the cell. |
| object_class                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | string  | The class of the object. This is a constant equal to <code>cell</code> .                                                                                                                                                                         |

*Table 12-2 Attributes of the cell Object Class (Continued)*

| Attribute name                | Type   | Description                                                                                                                                                                                                       |
|-------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| peak_power                    | double | The peak power of a cell in watts<br>This is the largest power value for that cell in the simulation waveform.                                                                                                    |
| peak_power_end_time           | double | The end time of the time interval in which peak power is measured for that cell. The unit of measurement is nanoseconds.                                                                                          |
| peak_power_start_time         | double | The start time of the time interval in which peak power is measured for that cell. The unit of measurement is nanoseconds.                                                                                        |
| power_states                  | double | The number of power state changes for the cell. This is the sum of the number of transitions on all input and output pins of the cell.                                                                            |
| ref_name                      | string | The name of the design or library cell of which the cell is (or will be) an instantiation. Also known as the reference name. The linker looks for a design or library cell by this name to resolve the reference. |
| switching_power               | double | The switching power of a cell in watts. It is the power dissipated by the charging and discharging of the load capacitance at the output of the cell.                                                             |
| switching_power_derate_factor | float  | Power derate factors, specified using the <code>set_power_derate</code> command, that apply to the cell.                                                                                                          |
| temperature_max               | float  | Maximum temperature specified for the cell through the operating condition specification or application of the <code>set_temperature</code> command.                                                              |

*Table 12-2 Attributes of the cell Object Class (Continued)*

| Attribute name                      | Type                | Description                                                                                                                                                                                                                                                                                                                                       |
|-------------------------------------|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>temperature_min</code>        | <code>float</code>  | Minimum temperature specified for the cell through the operating condition specification or application of the <code>set_temperature</code> command.                                                                                                                                                                                              |
| <code>timing_model_type</code>      | <code>string</code> | <p>Returns the timing model type of a cell. Valid values:</p> <ul style="list-style-type: none"> <li>• ITS (interface timing specification)</li> <li>• QTM (quick timing model)</li> <li>• <code>extracted</code></li> <li>• <code>none</code> (normal library model)</li> </ul> <p>You can set the <code>timing_model_type</code> attribute.</p> |
| <code>total_power</code>            | <code>double</code> | The total power of a cell in watts. It is the sum of dynamic power and leakage power.                                                                                                                                                                                                                                                             |
| <code>upf_isolation_strategy</code> | <code>string</code> | Strategy name (created by the <code>set_isolation</code> command) that was used to derive rail supply and ground nets of the cell.                                                                                                                                                                                                                |
| <code>upf_retention_strategy</code> | <code>string</code> | Strategy name (created by the <code>set_retention</code> command) that was used to derive backup rail supply and ground nets.                                                                                                                                                                                                                     |
| <code>wire_load_model_max</code>    | <code>string</code> | The name of the wire load model effective on a hierarchical cell for the maximum operating condition. You can set the <code>wire_load_model_max</code> attribute.                                                                                                                                                                                 |

*Table 12-2 Attributes of the cell Object Class (Continued)*

| Attribute name                | Type   | Description                                                                                                                                                                                                         |
|-------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wire_load_model_min           | string | The name of the wire load model effective on a hierarchical cell for the minimum operating condition (valid in bc_wc or on_chip_variation analysis types).<br>You can set the wire_load_model_min attribute.        |
| wire_load_selection_group_max | string | The name of the wire load selection group on a hierarchical cell for the maximum operating condition. You can set the wire_load_selection_group_max attribute.                                                      |
| wire_load_selection_group_min | string | The name of the wire load selection group on a hierarchical cell for the minimum operating condition (valid in bc_wc or on_chip_variation analysis types). You can set the wire_load_selection_group_min attribute. |

## Clock Object Class Attributes

[Table 12-3](#) lists the clock object class attributes.

*Table 12-3 Attributes of the clock Object Class*

| Attribute name         | Type  | Description                                                                                     |
|------------------------|-------|-------------------------------------------------------------------------------------------------|
| clock_latency_fall_max | float | The maximum fall latency (insertion delay) for a clock. Set with the set_clock_latency command. |
| clock_latency_fall_min | float | The minimum fall latency (insertion delay) for a clock. Set with the set_clock_latency command. |



*Table 12-3 Attributes of the clock Object Class (Continued)*

| Attribute name                                   | Type        | Description                                                                                                  |
|--------------------------------------------------|-------------|--------------------------------------------------------------------------------------------------------------|
| <code>clock_latency_rise_max</code>              | float       | The maximum rise latency (insertion delay) for a clock. Set with the <code>set_clock_latency</code> command. |
| <code>clock_latency_rise_min</code>              | float       | The minimum rise latency (insertion delay) for a clock. Set with the <code>set_clock_latency</code> command. |
| <code>clock_network_pins</code>                  | collections | The collection of pin and port objects that make up the propagation path of this clock.                      |
| <code>clock_source_latency_early_fall_max</code> | float       | The maximum early falling source latency. Set with the <code>set_clock_latency</code> command.               |
| <code>clock_source_latency_early_fall_min</code> | float       | The minimum early falling source latency. Set with the <code>set_clock_latency</code> command.               |
| <code>clock_source_latency_early_rise_max</code> | float       | The maximum early rising source latency. Set with the <code>set_clock_latency</code> command.                |
| <code>clock_source_latency_early_rise_min</code> | float       | The minimum early rising source latency. Set with the <code>set_clock_latency</code> command.                |
| <code>clock_source_latency_late_fall_max</code>  | float       | The maximum late falling source latency. Set with the <code>set_clock_latency</code> command.                |
| <code>clock_source_latency_late_fall_min</code>  | float       | The minimum late falling source latency. Set with the <code>set_clock_latency</code> command.                |
| <code>clock_source_latency_late_rise_max</code>  | float       | The maximum late rising source latency. Set with the <code>set_clock_latency</code> command.                 |
| <code>clock_source_latency_late_rise_min</code>  | float       | The minimum late rising source latency. Set with the <code>set_clock_latency</code> command.                 |

*Table 12-3 Attributes of the clock Object Class (Continued)*

| Attribute name                | Type                    | Description                                                                                                                                                                                                                                                                                            |
|-------------------------------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>full_name</code>        | <code>string</code>     | The name of the clock. This is set with the <code>create_clock</code> command. It is either the name given with the <code>-name</code> option, or the name of the first object to which the clock is attached. Once set, this attribute is read-only.                                                  |
| <code>generated_clocks</code> | <code>clock</code>      | This attribute is defined for any clock that is the parent of one or more generated clocks and returns those generated clock objects as a collection.                                                                                                                                                  |
| <code>hold_uncertainty</code> | <code>float</code>      | A floating-point value that specifies the clock uncertainty (skew) of a clock used for hold (and other minimum delay) timing checks. Set with the <code>set_clock_uncertainty</code> command.                                                                                                          |
| <code>is_active</code>        | <code>Boolean</code>    | This attribute is <code>true</code> if the clock is active (the default state). Clocks can be made active or inactive with <code>set_active_clocks</code> .                                                                                                                                            |
| <code>is_generated</code>     | <code>Boolean</code>    | This attribute is <code>true</code> if the clock is a generated clock. Set with the <code>create_generated_clock</code> command.                                                                                                                                                                       |
| <code>master_clock</code>     | <code>collection</code> | This attribute is defined for generated clock objects and returns the master clock object for that generated clock.                                                                                                                                                                                    |
| <code>master_pin</code>       | <code>collection</code> | This attribute is defined for generated clocks and returns the master source pin or port object used to determine the identity and polarity of the master clock. This corresponds to the pin or port provided with the <code>-source</code> option of the <code>create_generated_clock</code> command. |

*Table 12-3 Attributes of the clock Object Class (Continued)*

| Attribute name                               | Type  | Description                                                                                                                                                                                              |
|----------------------------------------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>max_capacitance_clock_path_fall</code> | float | A floating-point number that establishes an upper limit for the falling maximum capacitance for all pins in this clock path. Set with the <code>set_max_capacitance</code> command.                      |
| <code>max_capacitance_clock_path_rise</code> | float | A floating-point number that establishes an upper limit for the rising maximum capacitance for all pins in this clock path. Set with the <code>set_max_capacitance</code> command.                       |
| <code>max_capacitance_data_path_fall</code>  | float | A floating-point number that establishes an upper limit for the falling maximum capacitance for all pins in the data path launched by this clock. Set with the <code>set_max_capacitance</code> command. |
| <code>max_capacitance_data_path_rise</code>  | float | A floating-point number that establishes an upper limit for the rising maximum capacitance for all pins in the data path launched by this clock. Set with the <code>set_max_capacitance</code> command.  |
| <code>max_fall_delay</code>                  | float | A floating-point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_max_delay</code> command.                        |
| <code>max_rise_delay</code>                  | float | A floating-point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_max_delay</code> command.                         |

*Table 12-3 Attributes of the clock Object Class (Continued)*

| Attribute name                              | Type  | Description                                                                                                                                                                                                                                                            |
|---------------------------------------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>max_time_borrow</code>                | float | A floating-point number that establishes an upper limit for time borrowing; that is, it prevents the use of the entire pulse width for level-sensitive latches. Units are those used in the technology library. Set with the <code>set_max_time_borrow</code> command. |
| <code>max_transition_clock_path_fall</code> | float | A floating-point number that establishes an upper limit for the falling maximum transition for all pins in this clock path. Set with the <code>set_max_transition</code> command.                                                                                      |
| <code>max_transition_clock_path_rise</code> | float | A floating-point number that establishes an upper limit for the rising maximum transition for all pins in this clock path. Set with the <code>set_max_transition</code> command.                                                                                       |
| <code>max_transition_data_path_fall</code>  | float | A floating-point number that establishes an upper limit for the falling maximum transition for all pins in the data path launched by this clock. Set with the <code>set_max_transition</code> command.                                                                 |
| <code>max_transition_data_path_rise</code>  | float | A floating-point number that establishes an upper limit for the rising maximum transition for all pins in the data path launched by this clock. Set with the <code>set_max_transition</code> command.                                                                  |
| <code>min_fall_delay</code>                 | float | A floating-point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with <code>set_min_delay</code> .                                                                                                 |
| <code>min_rise_delay</code>                 | float | A floating-point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_min_delay</code> command.                                                                                       |

*Table 12-3 Attributes of the clock Object Class (Continued)*

| Attribute name                 | Type                    | Description                                                                                                                                                                                                                                                                      |
|--------------------------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>object_class</code>      | <code>string</code>     | The class of the object. This is a constant, equal to "clock". You cannot set the <code>object_class</code> attribute.                                                                                                                                                           |
| <code>period</code>            | <code>float</code>      | The clock period (or cycle time) is the shortest time during which the clock waveform repeats. For a simple waveform with one rising and one falling edge, the period is the difference between successive rising edges. Set with the <code>create_clock -period</code> command. |
| <code>propagated_clock</code>  | <code>Boolean</code>    | Specifies that clock latency (insertion delay) be determined by propagating delays from the clock source to destination register clock pins. If this attribute is not present, ideal clocking is assumed. Set with the <code>set_propagated_clock</code> command.                |
| <code>setup_uncertainty</code> | <code>float</code>      | A floating-point value that specifies the clock uncertainty (skew) of a clock used for setup (and other maximum delay) timing checks. Set with the <code>set_clock_uncertainty</code> command.                                                                                   |
| <code>sources</code>           | <code>collection</code> | This is a collection of the source pins or ports of the clock. The sources are defined with the <code>create_clock</code> command.                                                                                                                                               |
| <code>waveform</code>          | <code>string</code>     | This is a <code>string</code> representation of the clock waveform. For example, a clock rising at 2.5 and falling at 5.0 has a waveform attribute value of {2.5 5}. The waveform is defined with the <code>create_clock</code> command.                                         |

## Design Object Class Attributes

Table 12-4 lists the design object class attributes.

Table 12-4 Attributes of the design Object Class

| Attribute name            | Type    | Description                                                                                                                                                                                                                                                                                                                             |
|---------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| analysis_type             | string  | The analysis type: single, bc_wc, or on_chip_variation.                                                                                                                                                                                                                                                                                 |
| area                      | float   | The total area of the design. This is the sum of the areas of all leaf cells and nets.                                                                                                                                                                                                                                                  |
| capacitance_unit_in_farad | float   | Specifies the unit of capacitance in the main library in farads. This attribute is read-only; you cannot change the setting.                                                                                                                                                                                                            |
| current_unit_in_amp       | float   | Specifies the unit of capacitance in the main library in amps. This attribute is read-only; you cannot change the setting.                                                                                                                                                                                                              |
| designWare                | Boolean | This attribute specifies if a cell came from a DesignWare design.                                                                                                                                                                                                                                                                       |
| dont_touch                | Boolean | Identifies designs to be excluded from optimization in Design Compiler. Values are undefined by default. Designs with the dont_touch attribute set to true are not modified or replaced during compile in Design Compiler. Set with the set_dont_touch command and used by the characterize_context and create_timing_context commands. |
| dynamic_power             | double  | The dynamic power of the design in watts. It is the sum of the dynamic power of all the cells of the design.                                                                                                                                                                                                                            |

**Table 12-4** *Attributes of the design Object Class (Continued)*

| Attribute name                          | Type   | Description                                                                                                                                                                                                                                                                                                                          |
|-----------------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| early_cell_check_derate_factor          | float  | Early timing derate factors, specified using the <code>set_timing_derate</code> command, that apply to the design.                                                                                                                                                                                                                   |
| early_clk_cell_derate_factor            |        |                                                                                                                                                                                                                                                                                                                                      |
| early_clk_net_delta_derate_factor       |        |                                                                                                                                                                                                                                                                                                                                      |
| early_clk_net_derate_factor             |        |                                                                                                                                                                                                                                                                                                                                      |
| early_data_cell_derate_factor           |        |                                                                                                                                                                                                                                                                                                                                      |
| early_data_net_delta_derate_factor      |        |                                                                                                                                                                                                                                                                                                                                      |
| early_data_net_derate_factor            |        |                                                                                                                                                                                                                                                                                                                                      |
| early_fall_cell_check_derate_factor     |        |                                                                                                                                                                                                                                                                                                                                      |
| early_fall_clk_cell_derate_factor       |        |                                                                                                                                                                                                                                                                                                                                      |
| early_fall_clk_net_delta_derate_factor  |        |                                                                                                                                                                                                                                                                                                                                      |
| early_fall_clk_net_derate_factor        |        |                                                                                                                                                                                                                                                                                                                                      |
| early_fall_data_cell_derate_factor      |        |                                                                                                                                                                                                                                                                                                                                      |
| early_fall_data_net_delta_derate_factor |        |                                                                                                                                                                                                                                                                                                                                      |
| early_fall_data_net_derate_factor       |        |                                                                                                                                                                                                                                                                                                                                      |
| early_rise_cell_check_derate_factor     |        |                                                                                                                                                                                                                                                                                                                                      |
| early_rise_clk_cell_derate_factor       |        |                                                                                                                                                                                                                                                                                                                                      |
| early_rise_clk_net_delta_derate_factor  |        |                                                                                                                                                                                                                                                                                                                                      |
| early_rise_clk_net_derate_factor        |        |                                                                                                                                                                                                                                                                                                                                      |
| early_rise_data_cell_derate_factor      |        |                                                                                                                                                                                                                                                                                                                                      |
| early_rise_data_net_delta_derate_factor |        |                                                                                                                                                                                                                                                                                                                                      |
| early_rise_data_net_derate_factor       |        |                                                                                                                                                                                                                                                                                                                                      |
| extended_name                           | string | The complete, unambiguous name of a design. The <code>extended_name</code> of the design is the <code>source_file_name</code> attribute followed by a colon (:) followed by the <code>full_name</code> attribute. For example, the <code>extended_name</code> of design TOP read in from /u/user/simple.db is /u/user/simple.db:TOP. |
| full_name                               | string | The name of a design. For example, the <code>full_name</code> of design TOP read in from /u/user/simple.db is TOP. This name can be ambiguous because several designs of the same name can be read in from different files.                                                                                                          |
| gate_leakage_power                      | double | Gate leakage power of the design in watts. It is the sum of the gate leakage of all the cells of the design                                                                                                                                                                                                                          |

*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | Type    | Description                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>glitch_power</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | double  | Glitch power of the design in watts. It is the sum of the glitch power of all the cells of the design.                                                   |
| <code>internal_power</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | double  | Internal power of the design in watts. It is the sum of the internal power of all the cells of the design.                                               |
| <code>internal_power_derate_factor</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | float   | Power derate factors, specified using the <code>set_power_derate</code> command, that apply to the design.                                               |
| <code>intrinsic_leakage_power</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | double  | Intrinsic leakage power of the design in watts. It is the sum of the intrinsic leakage of all the cells of the design.                                   |
| <code>is_current</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Boolean | This attribute is <code>true</code> for the current design. This attribute changes for all designs when you use the <code>current_design</code> command. |
| <code>is_edited</code>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | Boolean | This attributed is <code>true</code> if the design has been uniquified as a result of a netlist editing (ECO) change.                                    |
| <code>late_cell_check_derate_factor</code><br><code>late_clk_cell_derate_factor</code><br><code>late_clk_net_delta_derate_factor</code><br><code>late_clk_net_derate_factor</code><br><code>late_data_cell_derate_factor</code><br><code>late_data_cell_derate_factor</code><br><code>late_data_net_delta_derate_factor</code><br><code>late_data_net_derate_factor</code><br><code>late_fall_cell_check_derate_factor</code><br><code>late_fall_clk_cell_derate_factor</code><br><code>late_fall_clk_net_delta_derate_factor</code><br><code>late_fall_clk_net_derate_factor</code><br><code>late_fall_data_cell_derate_factor</code><br><code>late_fall_data_net_delta_derate_factor</code><br><code>late_fall_data_net_derate_factor</code><br><code>late_rise_cell_check_derate_factor</code><br><code>late_rise_clk_cell_derate_factor</code><br><code>late_rise_clk_net_delta_derate_factor</code><br><code>late_rise_clk_net_derate_factor</code><br><code>late_rise_data_cell_derate_factor</code><br><code>late_rise_data_net_delta_derate_factor</code><br><code>late_rise_data_net_derate_factor</code> | float   | Late timing derate factors, specified using the <code>set_timing_derate</code> command, that apply to the design.                                        |



*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name                           | Type                | Description                                                                                                                                                                                                                                                                     |
|------------------------------------------|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>leakage_power</code>               | <code>double</code> | Leakage power of the design in watts. It is the sum of the leakage power of all the cells of the design.                                                                                                                                                                        |
| <code>leakage_power_derate_factor</code> | <code>float</code>  | Power derate factors, specified using the <code>set_power_derate</code> command, that apply to the design.                                                                                                                                                                      |
| <code>max_area</code>                    | <code>float</code>  | A floating-point number that represents the target area of the design. The units must be consistent with the units used from the technology library during optimization. The <code>max_area</code> value is set in Design Compiler using the <code>set_max_area</code> command. |
| <code>max_capacitance</code>             | <code>float</code>  | A floating-point number that sets the default maximum capacitance design rule limit for the design. The units must be consistent with those of the technology library used during optimization. Set with the <code>set_max_capacitance</code> command.                          |
| <code>max_fanout</code>                  | <code>float</code>  | A floating-point number that specifies the default maximum fanout design rule limit for the design. The units must be consistent with those of the technology library used during optimization. Set with the <code>set_max_fanout</code> command.                               |
| <code>max_transition</code>              | <code>float</code>  | A floating-point number that specifies the default maximum transition design rule limit for the design. The units must be consistent with those of the technology library used during optimization. Set with the <code>set_max_transition</code> command.                       |

*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name                       | Type   | Description                                                                                                                                                                                                                                               |
|--------------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>min_capacitance</code>         | float  | A floating-point number that sets the default minimum capacitance design rule limit for the design. The units must be consistent with those of the technology library used during optimization. Set with the <code>set_min_capacitance</code> command.    |
| <code>min_fanout</code>              | float  | A floating-point number that specifies the default minimum fanout design rule limit for the design. The units must be consistent with those of the technology library used during optimization. Set with the <code>set_min_fanout</code> command.         |
| <code>min_transition</code>          | float  | A floating-point number that specifies the default minimum transition design rule limit for the design. The units must be consistent with those of the technology library used during optimization. Set with the <code>set_min_transition</code> command. |
| <code>object_class</code>            | string | The class of the object. This is a constant equal to <code>design</code> .                                                                                                                                                                                |
| <code>operating_condition_max</code> | string | The name of the maximum or single operating condition for the design. Set with the <code>set_operating_conditions</code> command.                                                                                                                         |
| <code>operating_condition_min</code> | string | The name of the minimum operating condition for the design. This attribute is not valid in single operating condition analysis.                                                                                                                           |
| <code>peak_power</code>              | double | Peak power of the design in watts. The peak power of the design is not the sum of the peak power of the cells of the design.                                                                                                                              |

*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name                     | Type   | Description                                                                                                                                                                                                                                                                                          |
|------------------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>peak_power_end_time</code>   | double | The end time of the time interval in which peak power is measured for the design. The unit is nanoseconds (ns).                                                                                                                                                                                      |
| <code>peak_power_start_time</code> | double | Start time of the time interval in which peak power is measured for the design. The unit is nanoseconds (ns).                                                                                                                                                                                        |
| <code>power_simulation_time</code> | float  | Total simulation time in averaged mode. The unit is nanoseconds (ns).                                                                                                                                                                                                                                |
| <code>power_states</code>          | double | The sum of the <code>power_states</code> of all the cells of the design                                                                                                                                                                                                                              |
| <code>process_max</code>           | float  | The process value of the maximum or single operating condition for the design. The operating condition is defined in a library or by the <code>create_operating_conditions</code> command. You associate operating conditions with a design using the <code>set_operating_conditions</code> command. |
| <code>process_min</code>           | float  | The process value of the minimum operating condition for the design. The operating condition is defined in a library or by the <code>create_operating_conditions</code> command. Operating conditions are associated with a design using the <code>set_operating_conditions</code> command.          |

*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name                                                                                                                                                                                                                                                                                            | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                   |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rc_input_threshold_pct_rise<br>rc_input_threshold_pct_fall<br>rc_output_threshold_pct_rise<br>rc_output_threshold_pct_fall<br>rc_slew_derate_from_library<br>rc_slew_lower_threshold_pct_fall<br>rc_slew_lower_threshold_pct_rise<br>rc_slew_upper_threshold_pct_fall<br>rc_slew_upper_threshold_pct_rise | float  | The characterization trip points (waveform measurement thresholds) and slew derating factor that PrimeTime uses to calculate delays and transition times. These attributes from a design object return the values obtained from the main library (the first library in the link path). For more information, see <a href="#">“Characterization Trip Points” on page 9-6</a> .                 |
| resistance_unit_in_ohm                                                                                                                                                                                                                                                                                    | float  | Specifies the unit of resistance in the main library in farads. This attribute is read-only; you cannot change the setting.                                                                                                                                                                                                                                                                   |
| source_file_name                                                                                                                                                                                                                                                                                          | string | The name of the file from which the design was read. For example, the <code>source_file_name</code> of design TOP read in from /u/user/simple.db is /u/user/simple.db.                                                                                                                                                                                                                        |
| switching_power                                                                                                                                                                                                                                                                                           | double | Switching power of the design in watts. It is the sum of the switching power of all the cells of the design.                                                                                                                                                                                                                                                                                  |
| switching_power_derate_factor                                                                                                                                                                                                                                                                             | float  | Power derate factors, specified using the <code>set_power_derate</code> command, that apply to the design.                                                                                                                                                                                                                                                                                    |
| temperature_max                                                                                                                                                                                                                                                                                           | float  | The temperature value of the maximum or single operating condition for the design. The operating condition is defined in a library or by the <code>create_operating_conditions</code> command. You associate operating conditions with a design using the <code>set_operating_conditions</code> command. This attribute represents the ambient temperature value for the operating condition. |

*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name                   | Type                | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|----------------------------------|---------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>temperature_min</code>     | <code>float</code>  | The temperature value of the minimum operating condition for the design. The operating condition is defined in a library or by the <code>create_operating_conditions</code> command. You associate operating conditions with a design using the <code>set_operating_conditions</code> command. This attribute represents the ambient temperature value for the operating condition.                                                                                                                                                                                |
| <code>time_unit_in_second</code> | <code>float</code>  | Specifies the unit of time in the main library in farads. This attribute is read-only; you cannot change the setting.                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| <code>total_power</code>         | <code>double</code> | Total power of the design in watts. It is the sum of the total power of all the cells of the design.                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| <code>tree_type_max</code>       | <code>string</code> | The <code>tree_type</code> value of the maximum or single operating condition for the design. The operating condition is defined in a library or by the <code>create_operating_conditions</code> command. You associate operating conditions with a design using the <code>set_operating_conditions</code> command. The <code>tree_type</code> value is used in prelayout interconnect delay estimation, and can have a value of <code>best_case</code> , <code>balanced_case</code> , <code>balanced_resistance</code> (cmos2 only), or <code>worst_case</code> . |

*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name                    | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-----------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>tree_type_min</code>        | string | The <code>tree_type</code> value of the minimum operating condition for the design. The operating condition is defined in a library or by the <code>create_operating_conditions</code> command. You associate operating conditions with a design using the <code>set_operating_conditions</code> command. The <code>tree_type</code> value is used in prelayout interconnect delay estimation, and can have a value of <code>best_case</code> , <code>balanced_case</code> , <code>balanced_resistance</code> (cmos2 only), or <code>worst_case</code> . |
| <code>voltage_max</code>          | float  | The voltage value of the maximum or single operating condition for the design. The operating condition is defined in a library or by the <code>create_operating_conditions</code> command. You associate operating conditions with a design using the <code>set_operating_conditions</code> command. This attribute represents the supply voltage value for the operating condition.                                                                                                                                                                     |
| <code>voltage_min</code>          | float  | The voltage value of the minimum operating condition for the design. The operating condition is defined in a library or by the <code>create_operating_conditions</code> command. You associate operating conditions with a design using the <code>set_operating_conditions</code> command. This attribute represents the supply voltage value for the operating condition.                                                                                                                                                                               |
| <code>voltage_unit_in_volt</code> | float  | Specifies the unit of voltage in the main library in farads. This attribute is read-only; you cannot change the setting.                                                                                                                                                                                                                                                                                                                                                                                                                                 |

*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name           | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wire_load_min_block_size | float  | Specifies the smallest hierarchical cell that has automatic wire load selection by area applied. If an automatic wire load selection group is specified as the default in the main library, or through the <code>set_wire_load_selection_group</code> command, it is applied to all hierarchical cells larger than the specified minimum block size.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| wire_load_mode           | string | Determines which wire load model to use to compute wire capacitance, resistance, and area for nets in a hierarchical design that has different wire load models at different hierarchical levels. Allowed values are <code>top</code> , which tells PrimeTime to use the wire load model at the top hierarchical level; <code>enclosed</code> , which tells PrimeTime to use the wire load model on the smallest design that encloses a net completely; and <code>segmented</code> , which indicates to break the net into segments, one within each hierarchical level. In the segmented mode, each net segment is estimated using the wire load model on the design that encloses that segment. The segmented mode is not supported for wire load models on clusters. If no value is specified for this attribute, PrimeTime searches for a default in the first library in the link path. If no default is found, <code>top</code> is the default. Set with <code>set_wire_load_mode</code> . |
| wire_load_model_max      | string | The name of the design's wire load model for maximum conditions. Set with <code>set_wire_load_model</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| wire_load_model_min      | string | The name of the design's wire load model for minimum conditions. This attribute is not valid for single operating condition analysis. Set with <code>set_wire_load_model</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

*Table 12-4 Attributes of the design Object Class (Continued)*

| Attribute name                | Type   | Description                                                                                                                      |
|-------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------|
| wire_load_selection_group_max | string | The name of the design's wire load selection group for maximum conditions. Set with <code>set_wire_load_selection_group</code> . |
| wire_load_selection_group_min | string | The name of the design's wire load selection group for minimum conditions. Set with <code>set_wire_load_selection_group</code> . |

## Library Object Class Attributes

[Table 12-5](#) lists the library object class attributes.

*Table 12-5 Attributes of the lib Object Class*

| Attribute name            | Type   | Description                                                                                                                  |
|---------------------------|--------|------------------------------------------------------------------------------------------------------------------------------|
| capacitance_unit_in_farad | float  | Specifies the unit of capacitance in the main library in farads. This attribute is read-only; you cannot change the setting. |
| current_unit_in_amp       | string | Specifies the unit of capacitance in the main library in amps. This attribute is read-only; you cannot change the setting.   |
| default_connection_class  | string | Specifies the default connection class string for the connection class attribute of a pin or port.                           |
| default_max_capacitance   | float  | The library default maximum capacitance design rule limit.                                                                   |
| default_max_fanout        | float  | The library default maximum fanout design rule limit.                                                                        |
| default_max_transition    | float  | The library default maximum transition design rule limit.                                                                    |



*Table 12-5 Attributes of the lib Object Class (Continued)*

| Attribute name          | Type       | Description                                                                                                                                                                                                                                                                                                                                                                     |
|-------------------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| default_min_capacitance | float      | The library default minimum capacitance design rule limit.                                                                                                                                                                                                                                                                                                                      |
| default_min_fanout      | float      | The library default minimum fanout design rule limit.                                                                                                                                                                                                                                                                                                                           |
| default_min_transition  | float      | The library default minimum transition design rule limit.                                                                                                                                                                                                                                                                                                                       |
| extended_name           | string     | The complete, unambiguous name of a library. The <code>extended_name</code> of the library is the <code>source_file_name</code> attribute followed by a colon (:) followed by the <code>full_name</code> attribute. For example, the <code>extended_name</code> of library <code>tech1</code> read in from <code>/u/user/lib1.db</code> is <code>/u/user/lib1.db:tech1</code> . |
| full_name               | string     | The name of a library. For example, the <code>full_name</code> of library <code>tech1</code> read in from <code>/u/user/lib1.db</code> is <code>tech1</code> . This name can be ambiguous because several libraries of the same name can be read in from different files.                                                                                                       |
| has_sensitization_data  | Boolean    | The library has the sensitization data.                                                                                                                                                                                                                                                                                                                                         |
| lib_scaling_group       | collection | The collection of libraries in the scaling library group to which the library belongs, which is set with the <code>define_scaling_lib_group</code> command.                                                                                                                                                                                                                     |
| min_extended_name       | string     | Returns the full name (filename:lib_name) of the minimum library of the given library. This attribute is read-only; you cannot change the setting.                                                                                                                                                                                                                              |
| min_source_file_name    | string     | Returns the full name of the minimum library of the given library. This attribute is read-only; you cannot change the setting.                                                                                                                                                                                                                                                  |
| object_class            | string     | The class of the object. This is a constant equal to <code>lib</code> .                                                                                                                                                                                                                                                                                                         |

*Table 12-5 Attributes of the lib Object Class (Continued)*

| Attribute name         | Type   | Description                                                                                                                                               |
|------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------|
| resistance_unit_in_ohm | float  | Specifies the unit of resistance in the main library in farads. This attribute is read-only; you cannot change the setting.                               |
| source_file_name       | string | The name of the file from which the library was read. For example, the source_file_name of library tech1 read in from /u/user/lib1.db is /u/user/lib1.db. |
| time_unit_in_second    | float  | Specifies the unit of time in the main library in farads. This attribute is read-only; you cannot change the setting.                                     |
| voltage_unit_in_volt   | float  | Specifies the unit of voltage in the main library in farads. This attribute is read-only; you cannot change the setting.                                  |

## Library Cell Object Class Attributes

Table 12-6 lists the library cell object class attributes.

*Table 12-6 Attributes of the lib\_cell Object Class*

| Attribute name | Type    | Description                                                                                                                                                 |
|----------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|
| always_on      | Boolean | This attribute is <code>true</code> if the library cell is an always-on cell.                                                                               |
| area           | float   | A floating-point value representing the area of a library cell.                                                                                             |
| base_name      | string  | The name of a library cell. For example, the base_name of library cell tech1/AN2 is AN2.                                                                    |
| disable_timing | Boolean | This attribute is <code>true</code> if the timing for the library cell has been specified to be disabled using the <code>set_disable_timing</code> command. |

*Table 12-6 Attributes of the lib\_cell Object Class (Continued)*

| Attribute name                  | Type    | Description                                                                                                                                                                                                                           |
|---------------------------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| dont_touch                      | Boolean | Identifies library cells to be excluded from optimization. Values are undefined by default. Library cells with the dont_touch attribute set to true are not modified or replaced during compile. Set with the set_dont_touch command. |
| dont_use                        | Boolean | If this attribute exists on a library cell, it is excluded from the target library during optimization. Its setting is obtained from synthesis or place and route.                                                                    |
| full_name                       | string  | The fully qualified name of a library cell. This is the name of the library followed by the library cell name. For example, the full_name of library cell AN2 in library tech1 is tech1/AN2.                                          |
| function_id                     | string  | The name of the function that is created by Library Compiler.                                                                                                                                                                         |
| has_multi_ground_rails          | Boolean | This attribute is true if a library cell has multiple ground rails.                                                                                                                                                                   |
| has_multi_power_rails           | Boolean | This attribute is true if a library cell has multiple power rails.                                                                                                                                                                    |
| has_rail_specific_power_tables  | Boolean | This attribute is true if a library cell has multiple tables attached to rails.                                                                                                                                                       |
| is_black_box                    | Boolean | This attribute is true if the library cell is not sequential.                                                                                                                                                                         |
| is_combinational                | Boolean | This attribute is true if the library cell is not sequential.                                                                                                                                                                         |
| is_fall_edge_triggered          | Boolean | This attribute is true if library cell is used in the design as a falling-edge-triggered flip-flop.                                                                                                                                   |
| is_integrated_clock_gating_cell | Boolean | This attribute is true if the cell's reference is not linked to a library cell or design. This attribute is read-only; you cannot change the setting.                                                                                 |

*Table 12-6 Attributes of the lib\_cell Object Class (Continued)*

| Attribute name              | Type    | Description                                                                                                         |
|-----------------------------|---------|---------------------------------------------------------------------------------------------------------------------|
| is_isolation                | Boolean | This attribute is <code>true</code> if a library cell is an isolation cell.                                         |
| is_level_shifter            | Boolean | This attribute is <code>true</code> if the library cell is a level shifter cell.                                    |
| is_macro_switch             | Boolean | This attribute is <code>true</code> if the library cell is defined as a fine-grain switch cell in the library.      |
| is_mux                      | Boolean | This attribute is <code>true</code> if a library cell is a multiplexer.                                             |
| is_negative_level_sensitive | Boolean | This attribute is <code>true</code> if the library cell is used in the design as a negative level-sensitive latch.  |
| is_pad_cell                 | Boolean | This attribute is <code>true</code> if the library cell is a pad cell.                                              |
| is_pll_cell                 | Boolean | This attribute is <code>true</code> if the library cell is used in the design as a phase locked loop (PLL) cell.    |
| is_positive_level_sensitive | Boolean | This attribute is <code>true</code> if the library cell is used in the design as a positive level-sensitive latch.  |
| is_retention                | Boolean | This attribute is <code>true</code> if the library cell is a retention cell.                                        |
| is_rise_edge_triggered      | Boolean | This attribute is <code>true</code> if the library cell is used in the design as a rising-edge-triggered flip-flop. |
| is_sequential               | Boolean | This attribute is <code>true</code> if the library cell is sequential.                                              |
| is_three_state              | Boolean | This attribute is <code>true</code> if a library cell is a three-state device.                                      |
| mog_func_id                 | string  | The name of the cell's function that is created by Library Compiler for multiple output gate (MOG) cells.           |

*Table 12-6 Attributes of the lib\_cell Object Class (Continued)*

| Attribute name      | Type    | Description                                                                                                                                                                                                                                      |
|---------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| number_of_pins      | integer | Number of pins on the library cell.                                                                                                                                                                                                              |
| object_class        | string  | The class of the object. This is a constant, equal to <code>lib_cell</code> .                                                                                                                                                                    |
| timing_model_type   | string  | Returns the timing model type of a library cell. The values are <ul style="list-style-type: none"> <li>ITS (interface timing specification)</li> <li>QTM (quick timing model)</li> <li>extracted</li> <li>none (normal library model)</li> </ul> |
| user_function_class | string  | This attribute is used to size complex cells that would otherwise not be sizable due to limitations of library tools in generating the <code>function_id</code> attribute for those complex cells.                                               |

## Library Pin Object Class Attributes

[Table 12-7](#) lists the library pin object class attributes.

*Table 12-7 Attributes of the lib\_pin Object Class*

| Attribute name   | Type    | Description                                                                                                                              |
|------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------|
| base_name        | string  | The leaf name of the library cell pin. For example, the <code>base_name</code> of <code>tech1/AN2/Z</code> is <code>Z</code> .           |
| clock            | Boolean | A Boolean value that is set to <code>true</code> when a clock attribute is attached to a <code>lib_pin</code> in the library definition. |
| connection_class | string  | Specifies the connection class string for a pin.                                                                                         |
| direction        | string  | The direction of a pin. Value can be <code>in</code> , <code>out</code> , <code>inout</code> , or <code>internal</code> .                |

*Table 12-7 Attributes of the lib\_pin Object Class (Continued)*

| Attribute name                                                                                             | Type    | Description                                                                                                                                                                                                                                     |
|------------------------------------------------------------------------------------------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| disable_timing                                                                                             | Boolean | This attribute is <code>true</code> if the library pin has been specified to be disabled using the <code>set_disable_timing</code> command.                                                                                                     |
| drive_resistance_fall                                                                                      | float   | A floating-point value representing the linear drive resistance for falling delays of a library pin.                                                                                                                                            |
| drive_resistance_rise                                                                                      | float   | A floating-point value representing the linear drive resistance for rising delays of a library pin.                                                                                                                                             |
| driver_waveform_fall<br>driver_waveform_rise                                                               | string  | This attribute indicates the type of driver waveform for the library pin, either ramp or standard.                                                                                                                                              |
| fanout_load                                                                                                | float   | A floating-point value representing the fanout load value of a library pin. This value is used in computing <code>max_fanout</code> design rule cost.                                                                                           |
| full_name                                                                                                  | string  | The fully qualified name of a library cell pin. This is the name of the library followed by the library cell name followed by a pin name. For example, the <code>full_name</code> of pin Z on library cell AN2 in library tech1 is tech1/AN2/Z. |
| has_ccs_noise_above_high<br>has_ccs_noise_above_low<br>has_ccs_noise_below_high<br>has_ccs_noise_below_low | Boolean | This attribute indicates whether CCS noise information is present for a pin in a library.                                                                                                                                                       |
| has_ccs_receiver_fall                                                                                      | Boolean | This attribute indicates whether CCS information is present for a pin in a library, for receiver fall analysis.                                                                                                                                 |
| has_ccs_receiver_rise                                                                                      | Boolean | This attribute indicates whether CCS information is present for a pin in a library, for receiver rise analysis.                                                                                                                                 |
| is_async_pin                                                                                               | Boolean | This attribute is <code>true</code> if a library pin is an asynchronous preset/clear pin.                                                                                                                                                       |
| is_clear_pin                                                                                               | Boolean | This attribute is <code>true</code> if a library pin is an asynchronous clear pin.                                                                                                                                                              |

*Table 12-7 Attributes of the lib\_pin Object Class (Continued)*

| Attribute name                                     | Type    | Description                                                                                                                                                                       |
|----------------------------------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>is_clock_pin</code>                          | Boolean | This attribute is <code>true</code> if at least one instance of that clock pin exists in the design that has the <code>is_clock_pin</code> attribute equal to <code>true</code> . |
| <code>is_data_pin</code>                           | Boolean | This attribute is <code>true</code> if at least one instance of that data pin exists in the design that has the <code>is_data_pin</code> attribute equal to <code>true</code> .   |
| <code>is_fall_edge_triggered_clock_pin</code>      | Boolean | This attribute is <code>true</code> if the library pin is used in the design as a falling-edge-triggered flip-flop clock pin.                                                     |
| <code>is_fall_edge_triggered_data_pin</code>       | Boolean | This attribute is <code>true</code> if the library pin is used in the design as a falling-edge-triggered flip-flop data pin.                                                      |
| <code>is_mux_select_pin</code>                     | Boolean | This attribute is <code>true</code> if a library pin is a select pin of a multiplexer device.                                                                                     |
| <code>is_negative_level_sensitive_clock_pin</code> | Boolean | This attribute is <code>true</code> if the library pin is used in the design as a negative edge-triggered flip-flop clock pin.                                                    |
| <code>is_negative_level_sensitive_data_pin</code>  | Boolean | This attribute is <code>true</code> if the library pin is used in the design as a negative edge-triggered flip-flop data pin.                                                     |
| <code>is_pad</code>                                | Boolean | This attribute is <code>true</code> if the library pin is a pad. See the Library Compiler documentation.                                                                          |
| <code>is_pll_feedback_pin</code>                   | Boolean | This attribute is <code>true</code> if the library pin is a feedback pin of a phase locked loop (PLL) cell.                                                                       |
| <code>is_pll_output_pin</code>                     | Boolean | This attribute is <code>true</code> if the library pin is an output pin of a phase locked loop (PLL) cell.                                                                        |
| <code>is_pll_reference_pin</code>                  | Boolean | This attribute is <code>true</code> if the library pin is a reference pin of a phase locked loop (PLL) cell.                                                                      |

*Table 12-7 Attributes of the lib\_pin Object Class (Continued)*

| Attribute name                                     | Type    | Description                                                                                                                    |
|----------------------------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------|
| <code>is_positive_level_sensitive_clock_pin</code> | Boolean | This attribute is <code>true</code> if the library pin is used in the design as a positive edge-triggered flip-flop clock pin. |
| <code>is_positive_level_sensitive_data_pin</code>  | Boolean | This attribute is <code>true</code> if the library pin is used in the design as a positive edge-triggered flip-flop data pin.  |
| <code>is_preset_pin</code>                         | Boolean | This attribute is <code>true</code> if a library pin is an asynchronous preset pin.                                            |
| <code>is_rise_edge_triggered_clock_pin</code>      | Boolean | This attribute is <code>true</code> if the library pin is used in the design as a rising-edge-triggered flip-flop clock pin.   |
| <code>is_rise_edge_triggered_data_pin</code>       | Boolean | This attribute is <code>true</code> if the library pin is used in the design as a rising-edge-triggered flip-flop data pin.    |
| <code>is_three_state</code>                        | Boolean | This attribute is <code>true</code> if the library pin is a three-state driver.                                                |
| <code>is_three_state_enable_pin</code>             | Boolean | This attribute is <code>true</code> if a library pin is an enable pin of a three-state device.                                 |
| <code>is_three_state_output_pin</code>             | Boolean | This attribute is <code>true</code> if a library pin could output a three-state signal.                                        |
| <code>is_unbuffered</code>                         | Boolean | This attributed is <code>true</code> if the library pin is unbuffered. See the Library Compiler documentation.                 |
| <code>load_of_pin_capacitance</code>               | float   | The capacitance as specified by the Liberty file by the <code>capacitance</code> attribute in the pin object class.            |
| <code>max_capacitance</code>                       | float   | A floating-point value representing the maximum capacitance design rule limit for a library pin.                               |
| <code>max_fanout</code>                            | float   | A floating-point value representing the maximum fanout design rule limit for a library pin.                                    |



*Table 12-7 Attributes of the lib\_pin Object Class (Continued)*

| Attribute name           | Type   | Description                                                                                                                                   |
|--------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| max_transition           | float  | A floating-point value representing the maximum transition time design rule limit for a library pin.                                          |
| min_capacitance          | float  | A floating-point value representing the minimum capacitance design rule limit for a library pin.                                              |
| min_fanout               | float  | A floating-point value representing the minimum fanout design rule limit for a library pin.                                                   |
| min_transition           | float  | A floating-point value representing the minimum transition time design rule limit for a library pin.                                          |
| object_class             | string | The class of the object. This is a constant, equal to lib_pin.                                                                                |
| pin_capacitance          | float  | A floating-point value representing the capacitance of a library pin.                                                                         |
| pin_capacitance_max_fall | float  | Returns the maximum fall capacitance of the library pin. This attribute is read-only; you cannot change the setting.                          |
| pin_capacitance_max_rise | float  | Returns the maximum rise capacitance of the library pin. This attribute is read-only; you cannot change the setting.                          |
| pin_capacitance_min_fall | float  | Returns the minimum fall capacitance of the library pin. This attribute is read-only; you cannot change the setting.                          |
| pin_capacitance_min_rise | float  | Returns the minimum rise capacitance of the library pin. This attribute is read-only; you cannot change the setting.                          |
| pin_direction            | string | Specifies the direction of a pin. Allowed values are in, out, inout, or unknown. This attribute is read-only; you cannot change the settings. |

*Table 12-7 Attributes of the lib\_pin Object Class (Continued)*

| Attribute name                                                                                                                                                                                                                                                                                            | Type    | Description                                                                                                                                                                                                                                                                                                                                                |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rc_input_threshold_pct_rise<br>rc_input_threshold_pct_fall<br>rc_output_threshold_pct_rise<br>rc_output_threshold_pct_fall<br>rc_slew_derate_from_library<br>rc_slew_lower_threshold_pct_fall<br>rc_slew_lower_threshold_pct_rise<br>rc_slew_upper_threshold_pct_fall<br>rc_slew_upper_threshold_pct_rise | float   | The characterization trip points (waveform measurement thresholds) and slew derating factor that PrimeTime uses to calculate delays and transition times. These attributes from a library pin return the values obtained from the library to which the pin belongs. For more information, see <a href="#">“Characterization Trip Points” on page 9-6</a> . |
| si_has_immunity_above_high<br>si_has_immunity_above_low<br>si_has_immunity_below_high<br>si_has_immunity_below_low                                                                                                                                                                                        | Boolean | This attribute indicates whether NLDM noise immunity information is present for a pin in a library.                                                                                                                                                                                                                                                        |

## Library Timing Arc Object Class Attributes

[Table 12-8](#) lists the library timing arc object class attributes.

*Table 12-8 Attributes of the lib\_timing\_arc Object Class*

| Attribute name                                                                                             | Type       | Description                                                                                                                                                    |
|------------------------------------------------------------------------------------------------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| from_lib_pin                                                                                               | collection | A collection containing the from library pin of the library timing arc.                                                                                        |
| has_ccs_driver_fall                                                                                        | Boolean    | This attribute indicates whether the library timing arc has CCS Timing driver information, either <code>true</code> or <code>false</code> , for fall analysis. |
| has_ccs_driver_rise                                                                                        | Boolean    | This attribute indicates whether the library timing arc has CCS Timing driver information, either <code>true</code> or <code>false</code> , for rise analysis. |
| has_ccs_noise_above_low<br>has_ccs_noise_below_high<br>has_ccs_noise_below_low<br>has_ccs_noise_above_high | Boolean    | These attributes indicate whether CCS Noise information is present in the timing arc object in a library.                                                      |
| has_ccs_receiver_fall                                                                                      | Boolean    | This attribute indicates whether CCS receiver information is present in the timing arc object in a library, for fall analysis.                                 |

**Table 12-8** *Attributes of the lib\_timing\_arc Object Class (Continued)*

| Attribute name                                                                                                                 | Type    | Description                                                                                                                                                      |
|--------------------------------------------------------------------------------------------------------------------------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| has_ccs_receiver_rise                                                                                                          | Boolean | This attribute indicates whether CCS receiver information is present in the timing arc object in a library, for rise analysis.                                   |
| is_disabled                                                                                                                    | Boolean | Returns <code>true</code> if the library timing arc is disabled.                                                                                                 |
| is_user_disabled                                                                                                               | Boolean | Returns <code>true</code> if the library timing arc is disabled by using the <code>set_disable_timing</code> command,                                            |
| mode                                                                                                                           | string  | Returns the mode string of the library timing arc.                                                                                                               |
| object_class                                                                                                                   | string  | The class of the object. This is a constant equal to <code>library_arc</code> .                                                                                  |
| sdf_cond                                                                                                                       | string  | A string representing the SDF condition of the library timing arc.                                                                                               |
| sense                                                                                                                          | string  | A string representing the sense of the library timing arc.                                                                                                       |
| si_has_immunity_above_low<br>si_has_immunity_below_high<br>si_has_immunity_below_low<br>si_has_immunity_above_high             | Boolean | These attributes indicate whether noise immunity information is present in the timing arc object in a library.                                                   |
| si_has_iv_above_low<br>si_has_iv_below_high<br>si_has_iv_below_low<br>si_has_iv_above_high                                     | Boolean | These attributes indicate whether the library timing arc contains output steady-state information in the form of I/V relationships (polynomials or tables).      |
| si_has_propagation_above_low<br>si_has_propagation_below_high<br>si_has_propagation_below_low<br>si_has_propagation_above_high | Boolean | These attributes indicate whether the library timing arc contains information on how bumps present at the arc input are propagated across the arc to the output. |
| si_has_resistance_above_low<br>si_has_resistance_below_high<br>si_has_resistance_below_low<br>si_has_resistance_above_high     | Boolean | These attributes indicate whether the library timing arc contains output steady-state information in the form of simple steady drive resistance.                 |

*Table 12-8 Attributes of the lib\_timing\_arc Object Class (Continued)*

| Attribute name | Type       | Description                                                           |
|----------------|------------|-----------------------------------------------------------------------|
| to_lib_pin     | collection | A collection containing the to library pin of the library timing arc. |
| when           | string     | A string representing the when string of the library timing arc.      |

## Net Object Class Attributes

[Table 12-9](#) lists the net object class attributes.

*Table 12-9 Attributes of the net Object Class*

| Attribute name     | Type   | Description                                                                                                                                                                            |
|--------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| activity_source    | string | The source of switching activity information for the net. It is one of the following: file, set_switching_activity, set_case_analysis, propagated, implied, default, or UNINITIALIZED. |
| aggressors         | string | This attribute shows the aggressor nets that impact the victim net.                                                                                                                    |
| area               | float  | The estimated area of a net. The net area is calculated using a wire load model.                                                                                                       |
| ba_capacitance_max | float  | A floating-point value representing the back-annotated capacitance on a net for maximum conditions. Set with the set_load or read_parasitics command.                                  |
| ba_capacitance_min | float  | A floating-point value representing the back-annotated capacitance on a net for minimum conditions. Set with the set_load or read_parasitics command.                                  |

*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name                   | Type    | Description                                                                                                                                                                                                                                                                                                        |
|----------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ba_resistance_max</code>   | float   | A floating-point value representing the back-annotated resistance on a net for maximum conditions. Set with the <code>set_resistance</code> or <code>read_parasitics</code> command.                                                                                                                               |
| <code>ba_resistance_min</code>   | float   | A floating-point value representing the back-annotated resistance of a net for minimum conditions. Set with the <code>set_resistance</code> or <code>read_parasitics</code> command.                                                                                                                               |
| <code>base_name</code>           | string  | The leaf name of a net. For example, the base name of net <code>i1/i1z1</code> is <code>i1z1</code> . You cannot set this attribute.                                                                                                                                                                               |
| <code>coupling_capacitors</code> | string  | This attribute lists the cross-coupling capacitance values in pF. With this attribute the nets are explicitly identified. For example, <pre>{ula/A (n1) u2b/Z (n2) 0.40   not_filtered} {n1:3 (n1) n2:5 (n2) 0.03  filtered_by_accum_noise_peak} {in_port (n1) n3:7 (n3) 0.01  filtered_by_accum_noise_peak}</pre> |
| <code>dont_touch</code>          | Boolean | Identifies nets to be excluded from optimization in Design Compiler. Values are undefined by default. Nets with the <code>dont_touch</code> attribute set to true are not modified or replaced during compile with Design Compiler. Set with the <code>set_dont_touch</code> command.                              |

*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name                                                                                                                                                                                                                                                                                                           | Type   | Description                                                                                                                                                                                                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| early_fall_clk_net_delta_derate_factor<br>early_fall_clk_net_derate_factor<br>early_fall_data_net_delta_derate_factor<br>early_fall_data_net_derate_factor<br>early_rise_clk_net_delta_derate_factor<br>early_rise_clk_net_derate_factor<br>early_rise_data_net_delta_derate_factor<br>early_rise_data_net_derate_factor | float  | Early timing derate factors, specified using the <code>set_timing_derate</code> command, that apply to the net.                                                                                                                                                                                                |
| effective_aggressors                                                                                                                                                                                                                                                                                                     | string | This attribute lists the effective aggressors for the net. Effective aggressors are aggressors that are analyzed. For example, <code>get_attribute -class net n5 \ effective_aggressors n7</code> (For more information, see the <code>si_xtalk_bumps</code> attribute.)                                       |
| effective_coupling_capacitors                                                                                                                                                                                                                                                                                            | string | This attribute lists the effective cross-coupling capacitance values in pF. Only the capacitors that are not excluded are shown.                                                                                                                                                                               |
| escaped_full_name                                                                                                                                                                                                                                                                                                        | string | Contains the name of the cell. Any literal hierarchy characters are escaped with a backslash.                                                                                                                                                                                                                  |
| full_name                                                                                                                                                                                                                                                                                                                | string | The complete name of a net. For example, the <code>full_name</code> of net <code>i1z1</code> within cell <code>i1</code> is <code>i1/i1z1</code> . The <code>full_name</code> attribute is not affected by current instance. The <code>full_name</code> attribute is read-only; you cannot change the setting. |
| glitch_count                                                                                                                                                                                                                                                                                                             | float  | The number of glitch transitions on the net during the duration of <code>power_simulation_time</code> .                                                                                                                                                                                                        |
| glitch_rate                                                                                                                                                                                                                                                                                                              | double | The rate at which glitch transitions occur on the net. It is equal to <code>glitch_count/power_simulation_time</code> .                                                                                                                                                                                        |

*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name                                                                                                                                                                                                                                                                                                                                                                                                           | Type    | Description                                                                                                                                                                                                                                               |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>has_detailed_parasitics</code>                                                                                                                                                                                                                                                                                                                                                                                     | Boolean | This attribute is <code>true</code> if any part of the net has annotated detailed parasitics (even if only one segment of a net at different levels of hierarchy).                                                                                        |
| <code>has_valid_parasitics</code>                                                                                                                                                                                                                                                                                                                                                                                        | Boolean | This attribute is <code>true</code> if the net has an annotated pi model, or if all segments of the net are annotated with properly connected detailed parasitics that form a valid representation of physical interconnection between drivers and loads. |
| <code>is_design_mismatch</code>                                                                                                                                                                                                                                                                                                                                                                                          | Boolean | This attribute is <code>true</code> if an object is dirty, that is, the object is directly impacted by a mismatch between the block and top-level design.                                                                                                 |
| <code>is_ideal</code>                                                                                                                                                                                                                                                                                                                                                                                                    | Boolean | This attribute is <code>true</code> if the net has been marked ideal using the <code>set_ideal_network</code> command.                                                                                                                                    |
| <code>is_power_control_signal_net</code>                                                                                                                                                                                                                                                                                                                                                                                 | Boolean | This attribute is <code>true</code> if the signal net is used to control any power rail or PrimeTime PX rail mapping modes.                                                                                                                               |
| <code>late_fall_clk_net_delta_derate_factor</code><br><code>late_fall_clk_net_derate_factor</code><br><code>late_fall_data_net_delta_derate_factor</code><br><code>late_fall_data_net_derate_factor</code><br><code>late_rise_clk_net_delta_derate_factor</code><br><code>late_rise_clk_net_derate_factor</code><br><code>late_rise_data_net_delta_derate_factor</code><br><code>late_rise_data_net_derate_factor</code> | float   | Late timing derate factors, specified using the <code>set_timing_derate</code> command, that apply to the net.                                                                                                                                            |
| <code>net_resistance_max</code>                                                                                                                                                                                                                                                                                                                                                                                          | float   | A floating-point value representing the resistance of a net for maximum conditions. May be computed from wire load models or set using <code>set_resistance</code> or <code>read_parasitics</code> .                                                      |

*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name                          | Type    | Description                                                                                                                                                                                                                      |
|-----------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| net_resistance_min                      | float   | A floating-point value representing the resistance of a net for minimum conditions. May be computed from wire load models or set using <code>set_resistance</code> or <code>read_parasitics</code> .                             |
| number_of_aggressors                    | integer | This attribute shows the number of aggressor nets to a victim net. For example,<br><pre>get_attribute -class net n5 \     number_of_aggressors 1</pre> Note that any coupled net is an aggressor net.                            |
| number_of_coupling_capacitors           | integer | This attribute shows the number of coupling capacitors.                                                                                                                                                                          |
| number_of_effective_aggressors          | integer | This attribute shows the number of effective aggressor nets to a victim net. Only the effective aggressors are used for analysis. For example,<br><pre>get_attribute -class net n5 \     number_of_effective_aggr essors 1</pre> |
| number_of_effective_coupling_capacitors | integer | This attribute lists the number of effective coupling capacitors on a net. Only the effective coupling capacitors are used for the analysis.                                                                                     |
| object_class                            | string  | The class of the object. This is a constant, equal to <code>net</code> .                                                                                                                                                         |
| pin_capacitance_max                     | float   | A floating-point value representing the sum of all pin capacitances of a net for maximum conditions. You cannot set this attribute.                                                                                              |



*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name                        | Type    | Description                                                                                                                                                                                                                          |
|---------------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pin_capacitance_max_fall</code> | float   | Returns the maximum fall capacitance of all pins and ports for the net. This attribute is read-only; you cannot change the setting.                                                                                                  |
| <code>pin_capacitance_max_rise</code> | float   | Returns the maximum rise capacitance of all pins and ports for the net. This attribute is read-only; you cannot change the setting.                                                                                                  |
| <code>pin_capacitance_min</code>      | float   | A floating-point value representing the sum of all pin capacitances of a net for minimum conditions. You cannot set this attribute.                                                                                                  |
| <code>pin_capacitance_min_fall</code> | float   | Returns the minimum fall capacitance of all pins and ports for the net. This attribute is read-only; you cannot change the setting.                                                                                                  |
| <code>pin_capacitance_min_rise</code> | float   | Returns the minimum rise capacitance of all pins and ports for the net. This attribute is read-only; you cannot change the setting.                                                                                                  |
| <code>power_base_clock</code>         | string  | The name of the base clock associated with this net. If the net belongs to multiple clock domains, the <code>power_base_clock</code> attribute is set to the fastest of the clocks                                                   |
| <code>rc_annotated_segment</code>     | Boolean | This attribute is <code>true</code> if the specific net segment has annotated parasitics. For two segments at different levels of hierarchy (for example, <code>n1</code> and <code>h1/n1</code> ), the attribute values can differ. |

*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name                           | Type    | Description                                                                                                                                                                                                                                                                                                                                                                 |
|------------------------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>rc_network</code>                  | string  | A string describing the parasitic data that has been back-annotated on the net, including resistor values (in KOhms) and capacitor values (in pF).                                                                                                                                                                                                                          |
| <code>rc_network_with_sensitivity</code> | string  | Specifies the full RC network of a net with sensitivity values included. This attribute is read-only; you cannot change the setting.                                                                                                                                                                                                                                        |
| <code>si_double_switching_slack</code>   | float   | This net has double-switching slack on the victim net.                                                                                                                                                                                                                                                                                                                      |
| <code>si_has_double_switching</code>     | Boolean | This net has double-switching violation. Double-switching analysis needs to be enabled.                                                                                                                                                                                                                                                                                     |
| <code>si_is_selected</code>              | Boolean | This attribute exists on each coupled net. It indicates whether the net was reselected at least once ( <code>true</code> ) or was never reselected ( <code>false</code> ) for crosstalk analysis in the most recent timing update. Reselection can occur only in the second and subsequent iterations of crosstalk analysis.                                                |
| <code>si_xtalk_bumps</code>              | string  | This attribute lists each aggressor net and the voltage bumps that rising and falling aggressor transitions induce on the victim net (worst of rising minimum or maximum bumps and worst of falling minimum or maximum bumps, each expressed as a decimal fraction of the rail-to-rail voltage), or gives the reason that an aggressor net has no effect on the victim net. |

*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name                                                                                                                                                                                   | Type       | Description                                                                                                                                                                                                                                                                |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>si_xtalk_bumps_max_fall</code><br><code>si_xtalk_bumps_max_rise</code><br><code>si_xtalk_bumps_min_fall</code><br><code>si_xtalk_bumps_min_rise</code>                                     | string     | Each of these attributes lists each aggressor net and the voltage it induces on the victim net (expressed as a decimal fraction of the rail-to-rail voltage) for a given delay change type and transition type: maximum fall, maximum rise, minimum fall, or minimum rise. |
| <code>si_xtalk_composite_aggr_min_rise</code><br><code>si_xtalk_composite_aggr_min_fall</code><br><code>si_xtalk_composite_aggr_max_rise</code><br><code>si_xtalk_composite_aggr_max_fall</code> | collection | Each of these attributes list a collection of aggressors in a potential composite aggressor group for a given delay change type and transition type: maximum fall, maximum rise, minimum fall, or minimum rise.                                                            |
| <code>si_xtalk_used_ccs_min_rise</code><br><code>si_xtalk_used_ccs_min_fall</code><br><code>si_xtalk_used_ccs_max_rise</code><br><code>si_xtalk_used_ccs_max_fall</code>                         | Boolean    | Each attribute set to <code>true</code> means that the net was analyzed for crosstalk delay using CCS timing models for that type of timing constraint and transition.                                                                                                     |
| <code>static_probability</code>                                                                                                                                                                  | float      | The static probability of a net, which is the probability that the net has the logic value 1.                                                                                                                                                                              |
| <code>switching_power</code>                                                                                                                                                                     | double     | The switching power of a net in watts, which is the power dissipated by the charging and discharging of the capacitance of the net.                                                                                                                                        |
| <code>toggle_count</code>                                                                                                                                                                        | float      | The number of transitions of the net during the duration of the <code>power_simulation_time</code> attribute.                                                                                                                                                              |
| <code>toggle_rate</code>                                                                                                                                                                         | double     | The rate at which transitions occur on the net. It is equal to <code>toggle_count/power_simulation_time</code>                                                                                                                                                             |

*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name                                                                                                                                                                           | Type       | Description                                                                                                                                                                                                                         |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>total_capacitance_max</code>                                                                                                                                                       | float      | A floating-point value representing the sum of all pin capacitances and the wire capacitance of a net for maximum conditions. This attribute is read-only; you cannot change the setting.                                           |
| <code>total_capacitance_min</code>                                                                                                                                                       | float      | A floating-point value representing the sum of all pin capacitances and the wire capacitance of a net for minimum conditions. This attribute is read-only; you cannot change the setting.                                           |
| <code>total_ccs_capacitance_max_fall</code><br><code>total_ccs_capacitance_max_rise</code><br><code>total_ccs_capacitance_min_fall</code><br><code>total_ccs_capacitance_min_rise</code> | float      | This attribute lists the total capacitance of wire capacitance and CCS receiver capacitances. For example, <code>total_ccs_capacitance_max_rise</code> is the sum of wire capacitance and maximum of <code>rise_c1/rise_c2</code> . |
| <code>total_coupling_capacitance</code>                                                                                                                                                  | float      | This attribute lists the total cross-coupling capacitance a victim net has in pF.                                                                                                                                                   |
| <code>total_effective_coupling_capacitance</code>                                                                                                                                        | float      | This attribute lists the total effective cross capacitance a victim net has in pF. Only effective values are used during the analysis.                                                                                              |
| <code>user_global_coupling_separated</code>                                                                                                                                              | Boolean    | If <code>true</code> , this net has been globally separated with the <code>set_coupling_separation</code> command.                                                                                                                  |
| <code>user_pairwise_coupling_separated</code>                                                                                                                                            | collection | The collection of nets that have been pairwise-separated with the <code>set_coupling_separation</code> command.                                                                                                                     |

*Table 12-9 Attributes of the net Object Class (Continued)*

| Attribute name       | Type  | Description                                                                                                                                                                                                    |
|----------------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wire_capacitance_max | float | A floating-point value representing the wire capacitance of a net for maximum conditions. The value can be computed from wire load models or set using <code>set_load</code> or <code>read_parasitics</code> . |
| wire_capacitance_min | float | A floating-point value representing the wire capacitance of a net for minimum conditions. The value can be computed from wire load models or set using <code>set_load</code> or <code>read_parasitics</code> . |

## Path Group Object Class Attributes

[Table 12-10](#) lists the path group object class attributes.

*Table 12-10 Attributes of the path\_group Object Class*

| Attribute name | Type   | Description                                                                                                                                                                                                                                                                           |
|----------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| full_name      | string | The name of the path group. Path groups are created by <code>group_path</code> or implicitly using <code>create_clock</code> . This attribute is read-only; you cannot change the setting.                                                                                            |
| object_class   | string | The class of the object. This is a constant, equal to <code>path_group</code> . This attribute is read-only; you cannot change the setting.                                                                                                                                           |
| weight         | float  | A floating-point value representing the cost function weight assigned to this path group. The weight of a group specifies how much the group influences the total maximum delay cost and can be used to guide optimization. You can specify the weight with <code>group_path</code> . |

## Pin Object Class Attributes

[Table 12-11](#) lists the pin object class attributes.

*Table 12-11 Attributes of the pin Object Class*

| Attribute name                                   | Type   | Description                                                                                                                                                                                                                                                                |
|--------------------------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>actual_fall_transition_max</code>          | float  | A floating-point value representing the largest falling transition time for a pin.                                                                                                                                                                                         |
| <code>actual_fall_transition_min</code>          | float  | A floating-point value representing the smallest falling transition time for a pin.                                                                                                                                                                                        |
| <code>actual_min_clock_pulse_width_high</code>   | string | Returns a string containing a per-clock actual minimum pulse width value at a pin (high pulse).                                                                                                                                                                            |
| <code>actual_min_clock_pulse_width_low</code>    | string | Returns a string containing a per-clock actual minimum pulse width value at a pin (low pulse).                                                                                                                                                                             |
| <code>actual_rise_transition_max</code>          | float  | A floating-point value representing the largest rising transition time for a pin.                                                                                                                                                                                          |
| <code>actual_rise_transition_min</code>          | float  | A floating-point value representing the smallest rising transition time for a pin.                                                                                                                                                                                         |
| <code>annotated_fall_transition_delta_max</code> | float  | This attribute shows the additional transition time added to the maximum falling transition time on a pin. The additional transition time is set by either the <code>set_annotated_transition -delta_only</code> command or by PrimeTime SI during the crosstalk analysis. |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                                   | Type   | Description                                                                                                                                                                                                                                                                |
|--------------------------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>annotated_fall_transition_delta_min</code> | float  | This attribute shows the additional transition time added to the minimum falling transition time on a pin. The additional transition time is set by either the <code>set_annotated_transition -delta_only</code> command or by PrimeTime SI during the crosstalk analysis. |
| <code>annotated_rise_transition_delta_max</code> | float  | This attribute shows the additional transition time added to the maximum rising transition time on a pin. The additional transition time is set by either the <code>set_annotated_transition -delta_only</code> command or by PrimeTime SI during the crosstalk analysis.  |
| <code>annotated_rise_transition_delta_min</code> | float  | This attribute shows the additional transition time added to the minimum rising transition time on a pin. The additional transition time is set by either the <code>set_annotated_transition -delta_only</code> command or by PrimeTime SI during the crosstalk analysis.  |
| <code>arrival_window</code>                      | string | The minimum and maximum arrivals for rise and fall transitions. To get the <code>arrival_window</code> attribute on pins that are not endpoints, set the <code>timing_save_pin_arrival_and_slack</code> variable to true.                                                  |
| <code>cached_c1_max_fall</code>                  | float  | A floating-point value representing the C1 CCS receiver model for a maximum fall transition.                                                                                                                                                                               |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                                                                                                                                   | Type  | Description                                                                                                                                                                                                                                                              |
|--------------------------------------------------------------------------------------------------------------------------------------------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>cached_c1_max_rise</code>                                                                                                                  | float | A floating-point value representing the C1 CCS receiver model for a maximum rise transition.                                                                                                                                                                             |
| <code>cached_c1_min_fall</code>                                                                                                                  | float | A floating-point value representing the C1 CCS receiver model for a minimum fall transition.                                                                                                                                                                             |
| <code>cached_c1_min_rise</code>                                                                                                                  | float | A floating-point value representing the C1 CCS receiver model for a minimum rise transition.                                                                                                                                                                             |
| <code>cached_c2_max_fall</code>                                                                                                                  | float | A floating-point value representing the C2 CCS receiver model for a maximum fall transition.                                                                                                                                                                             |
| <code>cached_c2_max_rise</code>                                                                                                                  | float | A floating-point value representing the C2 CCS receiver model for a maximum rise transition.                                                                                                                                                                             |
| <code>cached_c2_min_fall</code>                                                                                                                  | float | A floating-point value representing the C2 CCS receiver model for a minimum fall transition.                                                                                                                                                                             |
| <code>cached_c2_min_rise</code>                                                                                                                  | float | A floating-point value representing the C2 CCS receiver model for a minimum rise transition.                                                                                                                                                                             |
| <code>cached_ceff_max_fall</code><br><code>cached_ceff_max_rise</code><br><code>cached_ceff_min_fall</code><br><code>cached_ceff_min_rise</code> | float | Worst effective capacitance of a net with detailed parasitics stored during the delay calculation that is connected to this output pin. This requires the <code>rc_cache_min_max_rise_fall_ceff</code> variable to be set to <code>true</code> before the timing update. |



*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                      | Type   | Description                                                                                                                                                                                                                                                                                                                                                                        |
|-------------------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>case_value</code>             | string | The user-specified logic value of a pin or port propagated from a case analysis or logic constant in the design. This attribute is computed only for leaf pins.                                                                                                                                                                                                                    |
| <code>ceff_params_max</code>        | string | Returns an access to the parameters internally computed by PrimeTime to compute the effective capacitance of a driver pin of a cell, respectively for maximum operating conditions. The returned parameters are <code>rd</code> , <code>t0</code> , <code>delta_t</code> , and <code>Ceff</code> ; they represent how a driver is modeled for computing the effective capacitance. |
| <code>ceff_params_min</code>        | string | Returns an access to the parameters internally computed by PrimeTime to compute the effective capacitance of a driver pin of a cell, respectively for minimum operating conditions. The returned parameters are <code>rd</code> , <code>t0</code> , <code>delta_t</code> , and <code>Ceff</code> ; they represent how a driver is modeled for computing the effective capacitance. |
| <code>clock_latency_fall_max</code> | float  | The user-specified maximum fall latency (insertion delay) of a pin in the clock network. Set with the <code>set_clock_latency</code> command.                                                                                                                                                                                                                                      |
| <code>clock_latency_fall_min</code> | float  | The user-specified minimum fall latency (insertion delay) of a pin in the clock network. Set with the <code>set_clock_latency</code> command.                                                                                                                                                                                                                                      |
| <code>clock_latency_rise_max</code> | float  | The user-specified maximum rise latency (insertion delay) of a pin in the clock network. Set with <code>set_clock_latency</code> .                                                                                                                                                                                                                                                 |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                                   | Type       | Description                                                                                                                                   |
|--------------------------------------------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------|
| <code>clock_latency_rise_min</code>              | float      | The user-specified minimum rise latency (insertion delay) of a pin in the clock network. Set with the <code>set_clock_latency</code> command. |
| <code>clock_source_latency_early_fall_max</code> | float      | The maximum early falling source latency. Set with the <code>set_clock_latency</code> command.                                                |
| <code>clock_source_latency_early_fall_min</code> | float      | The minimum early falling source latency. Set with the <code>set_clock_latency</code> command.                                                |
| <code>clock_source_latency_early_rise_max</code> | float      | The maximum early rising source latency. Set with the <code>set_clock_latency</code> command.                                                 |
| <code>clock_source_latency_early_rise_min</code> | float      | The minimum early rising source latency. Set with the <code>set_clock_latency</code> command.                                                 |
| <code>clock_source_latency_late_fall_max</code>  | float      | The maximum late falling source latency. Set with the <code>set_clock_latency</code> command.                                                 |
| <code>clock_source_latency_late_fall_min</code>  | float      | The minimum late falling source latency. Set with <code>set_clock_latency</code> .                                                            |
| <code>clock_source_latency_late_rise_max</code>  | float      | The maximum late rising source latency. Set with the <code>set_clock_latency</code> command.                                                  |
| <code>clock_source_latency_late_rise_min</code>  | float      | The minimum late rising source latency. Set with the <code>set_clock_latency</code> command.                                                  |
| <code>clocks</code>                              | collection | The collection of clock objects which propagate through this pin. It is undefined if no clocks are present.                                   |
| <code>constant_value</code>                      | string     | The logic value of a pin tied to logic constant zero or one in the netlist.                                                                   |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                             | Type       | Description                                                                                                                                                                                                               |
|--------------------------------------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>constraining_max_transition</code>   | float      | Returns the most constraining user-defined maximum transition value at a pin.                                                                                                                                             |
| <code>direction</code>                     | string     | The direction of a pin. Value can be in, out, inout, or internal. The <code>pin_direction</code> attribute is a synonym for direction. Directions can change as a result of linking a design, as references are resolved. |
| <code>disable_timing</code>                | Boolean    | Disables timing arcs. This has the same effect on timing as not having the arc in the library. Set with the <code>set_disable_timing</code> command.                                                                      |
| <code>driver_model_scaling_libs_max</code> | collection | The collection of library objects used for driver model scaling, where applicable, for libs maximum analysis.                                                                                                             |
| <code>driver_model_scaling_libs_min</code> | collection | The collection of library objects used for driver model scaling, where applicable, for libs minimum analysis.                                                                                                             |
| <code>driver_model_type_max_fall</code>    | string     | The driver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for maximum fall analysis.                                                                                                                       |
| <code>driver_model_type_max_rise</code>    | string     | The driver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for maximum rise analysis.                                                                                                                       |
| <code>driver_model_type_min_fall</code>    | string     | The driver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for minimum fall analysis.                                                                                                                       |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                          | Type   | Description                                                                                                                                                                                                                                                      |
|-----------------------------------------|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>driver_model_type_min_rise</code> | string | The driver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for minimum rise analysis.                                                                                                                                                              |
| <code>effective_capacitance_max</code>  | float  | The effective capacitance for maximum operating conditions. Valid only for driver pins attached to annotated RC networks.                                                                                                                                        |
| <code>effective_capacitance_min</code>  | float  | The effective capacitance for minimum operating conditions. Valid only for driver pins attached to annotated RC networks.                                                                                                                                        |
| <code>escaped_full_name</code>          | string | Contains the name of the cell. Any literal hierarchy characters are escaped with a backslash.                                                                                                                                                                    |
| <code>fanout_load</code>                | float  | A floating-point value representing the fanout load value of a pin. This value is used in computing max_fanout design rule cost.                                                                                                                                 |
| <code>full_name</code>                  | string | The complete name of a pin to the top of the hierarchy. For example, the full name of pin Z on cell U2 within cell U1 is U1/U2/Z. The setting of the current instance has no effect on the full name of a pin. See also the <code>lib_pin_name</code> attribute. |
| <code>glitch_rate</code>                | double | The rate at which the glitch transitions occur on the pin within a specific time period.                                                                                                                                                                         |
| <code>hold_uncertainty</code>           | float  | A floating-point value that specifies the clock uncertainty (skew) of a clock used for hold (and other minimum delay) timing checks. Set with the <code>set_clock_uncertainty</code> command.                                                                    |

**Table 12-11** *Attributes of the pin Object Class (Continued)*

| Attribute name                                                                                                   | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ideal_latency_max_fall<br>ideal_latency_max_rise<br>ideal_latency_min_fall<br>ideal_latency_min_rise             | float   | Ideal delay value annotated on a pin in an ideal network, using the <code>set_ideal_latency</code> command.                                                                                                                                                                                                                                                                                                                                                                                                        |
| ideal_transition_max_fall<br>ideal_transition_max_rise<br>ideal_transition_min_fall<br>ideal_transition_min_rise | float   | Ideal transition time annotated on a pin in an ideal network, using the <code>set_ideal_transition</code> command.                                                                                                                                                                                                                                                                                                                                                                                                 |
| is_async_pin                                                                                                     | Boolean | This attribute is <code>true</code> if a pin is an asynchronous preset/clear pin.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| is_clear_pin                                                                                                     | Boolean | This attribute is <code>true</code> if a pin is an asynchronous clear pin.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| is_clock_gating_pin                                                                                              | Boolean | This attribute is <code>true</code> if a pin is a pin of a clock-gating cell.                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| is_clock_pin                                                                                                     | Boolean | For instance <code>pin</code> objects, this attribute is <code>true</code> if the pin is a valid and active clock pin in the design that is reached by a clock signal and where that sequential cell is not disabled by disabled timing arcs or case analysis.                                                                                                                                                                                                                                                     |
| is_clock_used_as_clock<br>is_clock_used_as_data                                                                  | Boolean | When both are <code>true</code> , the clock through the specified pin acts as both clock and data. When <code>is_clock_used_as_clock</code> is <code>true</code> and <code>is_clock_used_as_data</code> is <code>false</code> , the clock through the specified pin acts only as a clock and never as data. When <code>is_clock_used_as_data</code> is <code>true</code> and <code>is_clock_used_as_clock</code> is <code>false</code> , the clock through the specified pin acts only as data and never as clock. |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                                | Type    | Description                                                                                                                                                                                                                                                                                                                                                  |
|-----------------------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>is_data_pin</code>                      | Boolean | <p>This attribute is <code>true</code> if a pin is a data pin of a sequential cell.</p> <p>For instance <code>pin</code> objects, this attribute is <code>true</code> if the pin is a valid and active data pin in the design that is reached by a clock signal and where that sequential cell is not disabled by disabled timing arcs or case analysis.</p> |
| <code>is_driver_scaled_max_fall</code>        | Boolean | Specifies whether the pin uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for maximum fall analysis                                                                                                                                                                                                    |
| <code>is_driver_scaled_max_rise</code>        | Boolean | Specifies whether the pin uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for maximum rise analysis.                                                                                                                                                                                                   |
| <code>is_driver_scaled_min_fall</code>        | Boolean | Specifies whether the pin uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for minimum fall analysis.                                                                                                                                                                                                   |
| <code>is_driver_scaled_min_rise</code>        | Boolean | Specifies whether the pin uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for minimum rise analysis.                                                                                                                                                                                                   |
| <code>is_fall_edge_triggered_clock_pin</code> | Boolean | This attribute is <code>true</code> if a pin is used in the design as a falling-edge-triggered flip-flop clock pin.                                                                                                                                                                                                                                          |
| <code>is_fall_edge_triggered_data_pin</code>  | Boolean | This attribute is <code>true</code> if a pin is used in the design as a falling-edge-triggered flip-flop data pin.                                                                                                                                                                                                                                           |
| <code>is_hierarchical</code>                  | Boolean | This attribute is <code>true</code> for any pins that are instantiations of another design, and <code>false</code> for pins that are instantiations of a library pin (also known as leaf pins).                                                                                                                                                              |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                                     | Type    | Description                                                                                                                                                     |
|----------------------------------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>is_design_mismatch</code>                    | Boolean | This attribute is <code>true</code> if an object is dirty, that is, the object is directly impacted by a mismatch between the block and top-level design.       |
| <code>is_ideal</code>                              | Boolean | This attribute is <code>true</code> if the pin has been marked ideal using the <code>set_ideal_network</code> command.                                          |
| <code>is_interface_logic_pin</code>                | Boolean | This attribute is <code>true</code> if the pin is in the interface logic model (ILM) of the design. This attribute is read-only; you cannot change the setting. |
| <code>is_mux_select_pin</code>                     | Boolean | This attribute is <code>true</code> if a pin is the select pin of a multiplexer device.                                                                         |
| <code>is_negative_level_sensitive_clock_pin</code> | Boolean | This attribute is <code>true</code> if a pin is used in the design as a negative edge-triggered flip-flop clock pin.                                            |
| <code>is_negative_level_sensitive_data_pin</code>  | Boolean | This attribute is <code>true</code> if a pin is used in the design as a negative edge-triggered flip-flop data pin.                                             |
| <code>is_port</code>                               | Boolean | This attribute is <code>true</code> for a pin or a port. Pins of ports are accessible only from a <code>timing_point</code> object.                             |
| <code>is_positive_level_sensitive_clock_pin</code> | Boolean | This attribute is <code>true</code> if a pin is used in the design as a positive edge-triggered flip-flop clock pin.                                            |
| <code>is_positive_level_sensitive_data_pin</code>  | Boolean | This attribute is <code>true</code> if a pin is used in the design as a positive edge-triggered flip-flop data pin.                                             |
| <code>is_preset_pin</code>                         | Boolean | This attribute is <code>true</code> if a pin is an asynchronous preset pin.                                                                                     |
| <code>is_receiver_scaled_max_fall</code>           | Boolean | Specifies whether the pin uses scaling between libraries for the receiver model, either <code>true</code> or <code>false</code> , for maximum fall analysis.    |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                                | Type    | Description                                                                                                                                                  |
|-----------------------------------------------|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>is_receiver_scaled_max_rise</code>      | Boolean | Specifies whether the pin uses scaling between libraries for the receiver model, either <code>true</code> or <code>false</code> , for maximum rise analysis. |
| <code>is_receiver_scaled_min_fall</code>      | Boolean | Specifies whether the pin uses scaling between libraries for the receiver model, either <code>true</code> or <code>false</code> , for minimum fall analysis. |
| <code>is_receiver_scaled_min_rise</code>      | Boolean | Specifies whether the pin uses scaling between libraries for the receiver model, either <code>true</code> or <code>false</code> , for minimum rise analysis. |
| <code>is_rise_edge_triggered_clock_pin</code> | Boolean | This attribute is <code>true</code> if a pin is used in the design as a rising-edge-triggered flip-flop clock pin.                                           |
| <code>is_rise_edge_triggered_data_pin</code>  | Boolean | This attribute is <code>true</code> if a pin is used in the design as a rising-edge-triggered flip-flop data pin.                                            |
| <code>is_three_state</code>                   | Boolean | This attribute is <code>true</code> if a pin is a three-state driver.                                                                                        |
| <code>is_three_state_enable_pin</code>        | Boolean | This attribute is <code>true</code> if a pin is an enable pin of a three-state device.                                                                       |
| <code>is_three_state_output_pin</code>        | Boolean | This attribute is <code>true</code> if a pin could output a three-state signal.                                                                              |
| <code>lib_pin_name</code>                     | string  | The leaf pin name. For example, the <code>lib_pin_name</code> of pin U2/U1/Z is Z. This attribute is read-only.                                              |
| <code>max_capacitance</code>                  | float   | A floating-point value representing the maximum capacitance design rule limit for a pin.                                                                     |



*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                | Type  | Description                                                                                                                                                                                                      |
|-------------------------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>max_fall_arrival</code> | float | A floating-point value that specifies the arrival time for the longest path with a falling transition on a pin. In best-case, worst-case mode, this value is for the worst-case operating condition.             |
| <code>max_fall_delay</code>   | float | A floating-point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_max_delay</code> command.                                |
| <code>max_fall_slack</code>   | float | A floating-point value representing the worst slack at a pin for falling maximum path delays. This attribute is valid for path endpoints (register data pins and primary outputs) after timing has been updated. |
| <code>max_fanout</code>       | float | A floating-point value representing the maximum fanout design rule limit for a pin.                                                                                                                              |
| <code>max_rise_arrival</code> | float | A floating-point value that specifies the arrival time for the longest path with a rising transition on a pin. In best-case, worst-case mode, this value is for the worst-case operating condition.              |
| <code>max_rise_delay</code>   | float | A floating-point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_max_delay</code> command.                                 |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                | Type  | Description                                                                                                                                                                                                                                                            |
|-------------------------------|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>max_rise_slack</code>   | float | A floating-point value representing the worst slack at a pin for rising maximum path delays. This attribute is valid for path endpoints (register data pins and primary outputs) after timing has been updated.                                                        |
| <code>max_time_borrow</code>  | float | A floating-point number that establishes an upper limit for time borrowing; that is, it prevents the use of the entire pulse width for level-sensitive latches. Units are those used in the technology library. Set with the <code>set_max_time_borrow</code> command. |
| <code>max_transition</code>   | float | A floating-point value representing the maximum transition time design rule limit for a pin.                                                                                                                                                                           |
| <code>min_capacitance</code>  | float | A floating-point value representing the minimum capacitance design rule limit for a pin.                                                                                                                                                                               |
| <code>min_fall_arrival</code> | float | A floating-point value that specifies the arrival time for the shortest path with a falling transition on a pin. In best-case worst-case mode, this value is for the best-case mode.                                                                                   |
| <code>min_fall_delay</code>   | float | A floating-point value that specifies the minimum falling delay on clocks, pins, cells, or on paths between such objects. Set with the <code>set_min_delay</code> command.                                                                                             |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                | Type   | Description                                                                                                                                                                                                                                                                            |
|-------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>min_fall_slack</code>   | float  | A floating-point value representing the worst slack at a pin for falling minimum path delays. This attribute is valid for path endpoints (register data pins and primary outputs) after timing is updated.                                                                             |
| <code>min_fanout</code>       | float  | A floating-point value representing the minimum fanout design rule limit for a pin.                                                                                                                                                                                                    |
| <code>min_rise_arrival</code> | float  | A floating-point value that specifies the worst hold slack of all paths passing through a pin with a rising transition at a pin. In best-case, worst-case operating conditions, this value is for the best-case condition. If all such paths are unconstrained, the value is infinity. |
| <code>min_rise_delay</code>   | float  | A floating-point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_min_delay</code> command.                                                                                                       |
| <code>min_rise_slack</code>   | float  | A floating-point value representing the worst slack at a pin for rising minimum path delays. This attribute is valid for path endpoints (register data pins and primary outputs) after timing has been updated.                                                                        |
| <code>min_transition</code>   | float  | A floating-point value representing the minimum transition time design rule limit for a pin.                                                                                                                                                                                           |
| <code>object_class</code>     | string | The class of the object. This is a constant, equal to pin.                                                                                                                                                                                                                             |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                        | Type   | Description                                                                                                                                                                               |
|---------------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pin_capacitance_max</code>      | float  | A floating-point value representing the capacitance of a pin for maximum conditions.                                                                                                      |
| <code>pin_capacitance_max_fall</code> | float  | Returns the maximum fall capacitance of the pin. This attribute is read-only; you cannot change the setting.                                                                              |
| <code>pin_capacitance_max_rise</code> | float  | Returns the maximum rise capacitance of the pin. This attribute is read-only; you cannot change the setting.                                                                              |
| <code>pin_capacitance_min</code>      | float  | A floating-point value representing the capacitance of a pin for minimum conditions.                                                                                                      |
| <code>pin_capacitance_min_fall</code> | float  | Returns the minimum fall capacitance of the pin. This attribute is read-only; you cannot change the setting.                                                                              |
| <code>pin_capacitance_min_rise</code> | float  | Returns the minimum rise capacitance of the pin. This attribute is read-only; you cannot change the setting.                                                                              |
| <code>pin_direction</code>            | string | The direction of a pin. Value can be in, out, inout, or internal. This attribute exists for backward compatibility with <code>dc_shell</code> . See the <code>direction</code> attribute. |
| <code>power_base_clock</code>         | string | The name of the base clock associated with this pin. If the pin belongs to multiple clock domains, the <code>power_base_clock</code> attribute is set to the fastest of the clocks.       |

Table 12-11 Attributes of the pin Object Class (Continued)

| Attribute name                                                                                                                                                                                                                                                                                                                                | Type       | Description                                                                                                                                                                                                                                                                                                                                                                       |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| power_rail_voltage_min<br>power_rail_voltage_max<br>power_rail_bidir_input_min<br>power_rail_bidir_input_max                                                                                                                                                                                                                                  | float      | Specifies the minimum or maximum power rail voltage set on the pin. In the case of a bidirectional pin, the output voltage is returned by default. To access the input voltage of a bidirectional pin, use the <code>_bidir_input_</code> attributes.                                                                                                                             |
| propagated_clock                                                                                                                                                                                                                                                                                                                              | Boolean    | Specifies that the clock edge times be delayed by propagating the values through the clock network. If this attribute is not present, ideal clocking is assumed. Set with <code>set_propagated_clock</code> .                                                                                                                                                                     |
| rc_input_threshold_pct_fall_max<br>rc_input_threshold_pct_rise_max<br>rc_output_threshold_pct_fall_max<br>rc_output_threshold_pct_rise_max<br>rc_slew_derate_from_library_max<br>rc_slew_lower_threshold_pct_rise_max<br>rc_slew_upper_threshold_pct_rise_max<br>rc_slew_lower_threshold_pct_fall_max<br>rc_slew_upper_threshold_pct_fall_max | float      | The characterization trip points (waveform measurement thresholds) and slew derating factor that PrimeTime uses to calculate delays and transition times. These attributes on an instance pin return the library threshold values obtained from the maximum library in a min-max analysis. For more information, see <a href="#">“Characterization Trip Points” on page 9-6</a> . |
| rc_input_threshold_pct_rise_min<br>rc_input_threshold_pct_fall_min<br>rc_output_threshold_pct_rise_min<br>rc_output_threshold_pct_fall_min<br>rc_slew_derate_from_library_min<br>rc_slew_lower_threshold_pct_rise_min<br>rc_slew_upper_threshold_pct_rise_min<br>rc_slew_lower_threshold_pct_fall_min<br>rc_slew_upper_threshold_pct_fall_min | float      | The characterization trip points (waveform measurement thresholds) and slew derating factor that PrimeTime uses to calculate delays and transition times. These attributes on an instance pin return the library threshold values obtained from the minimum library in a min-max analysis. For more information, see <a href="#">“Characterization Trip Points” on page 9-6</a> . |
| receiver_model_scaling_libs_max                                                                                                                                                                                                                                                                                                               | collection | The collection of library objects used for receiver model scaling, where applicable, for libs maximum analysis.                                                                                                                                                                                                                                                                   |

*Table 12-11 Attributes of the pin Object Class (Continued)*

| Attribute name                                                                                                                                                                                                     | Type       | Description                                                                                                                                                                         |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>receiver_model_scaling_libs_min</code>                                                                                                                                                                       | collection | The collection of library objects used for receiver model scaling, where applicable, for libs minimum analysis.                                                                     |
| <code>receiver_model_type_max_fall</code>                                                                                                                                                                          | string     | The receiver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for type maximum fall analysis.                                                                          |
| <code>receiver_model_type_max_rise</code>                                                                                                                                                                          | string     | The receiver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for type maximum rise analysis.                                                                          |
| <code>receiver_model_type_min_fall</code>                                                                                                                                                                          | string     | The receiver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for type minimum fall analysis.                                                                          |
| <code>receiver_model_type_min_rise</code>                                                                                                                                                                          | string     | The receiver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for type minimum rise analysis.                                                                          |
| <code>setup_uncertainty</code>                                                                                                                                                                                     | float      | A floating-point value that specifies the clock uncertainty (skew) of a clock used for setup (and other maximum delay) timing checks. Set with <code>set_clock_uncertainty</code> . |
| <code>si_noise_active_aggressors_above_high</code><br><code>si_noise_active_aggressors_above_low</code><br><code>si_noise_active_aggressors_below_high</code><br><code>si_noise_active_aggressors_below_low</code> | collection | Collection of a subset of effective aggressors that contributed to the worst case alignment to have largest impact on noise bump height.                                            |

Table 12-11 Attributes of the pin Object Class (Continued)

| Attribute name                                                                                                                                 | Type   | Description                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|------------------------------------------------------------------------------------------------------------------------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| si_noise_bumps_above_high<br>si_noise_bumps_below_high<br>si_noise_bumps_above_low<br>si_noise_bumps_below_low                                 | string | <p>This attribute returns a string that lists the aggressor nets for the input pin and their corresponding coupled bump heights and widths, considering noise bumps in the above-high, below-high, above-low, or below-low region. The format of the string is:</p> <pre>           {{aggr1_name height width}            {aggr2_name height width}            ... }         </pre> <p>Height is in volts and width is in library time units.</p> |
| si_noise_height_factor_above_high<br>si_noise_height_factor_above_low<br>si_noise_height_factor_below_high<br>si_noise_height_factor_below_low | float  | <p>This attribute returns the noise height derating factor for the specified pins, in the above-high, below-high, above-low, or below-low region.</p>                                                                                                                                                                                                                                                                                             |
| si_noise_height_offset_above_high<br>si_noise_height_offset_above_low<br>si_noise_height_offset_below_high<br>si_noise_height_offset_below_low | float  | <p>This attribute returns the noise height derating factor for the specified pins, in the above-high, below-high, above-low, or below-low region.</p>                                                                                                                                                                                                                                                                                             |
| si_noise_lib_pin_name                                                                                                                          | string | <p>Name of the library pin of equivalent library cell specified for noise analysis.</p>                                                                                                                                                                                                                                                                                                                                                           |
| si_noise_prop_bumps_above_high<br>si_noise_prop_bumps_below_high<br>si_noise_prop_bumps_above_low<br>si_noise_prop_bumps_below_low             | string | <p>This attribute returns a string that shows the bump height and width at the input pin caused by noise propagation, in the above-high, below-high, above-low, or below-low region. The format of the string is:</p> <pre>           {height width}         </pre> <p>Height is in volts and width is in library time units.</p>                                                                                                                 |
| si_noise_slack_above_high<br>si_noise_slack_below_high<br>si_noise_slack_above_low<br>si_noise_slack_below_low                                 | float  | <p>This attribute returns the amount of noise slack for the pin in the above-high, below-high, above-low, or below-low region.</p>                                                                                                                                                                                                                                                                                                                |

**Table 12-11** *Attributes of the pin Object Class (Continued)*

| Attribute name                                                                                                                                     | Type       | Description                                                                                                                                                                                                                                                                                                    |
|----------------------------------------------------------------------------------------------------------------------------------------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| si_noise_total_bump_above_high<br>si_noise_total_bump_above_low<br>si_noise_total_bump_below_high<br>si_noise_total_bump_below_low                 | string     | This attribute returns a string that shows the total bump height and width at the input pin caused by crosstalk and noise propagation, in the above-high, below-high, above-low, or below-low region. The format of the string is:<br>{height width}<br>Height is in volts and width is in library time units. |
| si_noise_width_factor_above_high<br>si_noise_width_factor_above_low<br>si_noise_width_factor_below_high<br>si_noise_width_factor_below_low         | float      | This attribute returns the noise width derating factor for the specified pins, in the above-high, below-high, above-low, or below-low region.                                                                                                                                                                  |
| si_noise_worst_prop_arc_above_high<br>si_noise_worst_prop_arc_above_low<br>si_noise_worst_prop_arc_below_high<br>si_noise_worst_prop_arc_below_low | collection | The cell arc that corresponds to the worst noise propagation to an output pin of a cell.                                                                                                                                                                                                                       |
| static_probability                                                                                                                                 | float      | The static probability of a pin. It is the probability that the pin has the logic value 1.                                                                                                                                                                                                                     |
| temperature_max<br>temperature_min                                                                                                                 | float      | Maximum or minimum temperature specified for the cell through the operating condition specification or application of the <code>set_temperature</code> command.                                                                                                                                                |
| toggle_rate                                                                                                                                        | double     | The rate at which transitions occur on the pin within a time period.                                                                                                                                                                                                                                           |
| user_case_value                                                                                                                                    | string     | The user-specified logic value of a pin or port.                                                                                                                                                                                                                                                               |



## Port Object Class Attributes

Table 12-12 lists the port object class attributes.

*Table 12-12 Attributes of the port Object Class*

| Attribute name                                   | Type   | Description                                                                                                                                                                                                                                                                 |
|--------------------------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>actual_fall_transition_max</code>          | float  | A floating-point value representing the largest falling transition time for a port. You cannot set this attribute.                                                                                                                                                          |
| <code>actual_fall_transition_min</code>          | float  | A floating-point value representing the smallest falling transition time for a port. You cannot set this attribute.                                                                                                                                                         |
| <code>actual_min_clock_pulse_width_high</code>   | string | Returns a string containing a per-clock actual minimum pulse width value at a port (high pulse).                                                                                                                                                                            |
| <code>actual_min_clock_pulse_width_low</code>    | string | Returns a string containing a per-clock actual minimum pulse width value at a port (low pulse).                                                                                                                                                                             |
| <code>actual_rise_transition_max</code>          | float  | A floating-point value representing the largest rising transition time for a port. You cannot set this attribute.                                                                                                                                                           |
| <code>actual_rise_transition_min</code>          | float  | A floating-point value representing the smallest rising transition time for a port. You cannot set this attribute.                                                                                                                                                          |
| <code>annotated_fall_transition_delta_max</code> | float  | This attribute shows the additional transition time added to the maximum falling transition time on a port. The additional transition time is set by either the <code>set_annotated_transition -delta_only</code> command or by PrimeTime SI during the crosstalk analysis. |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                                                                                                                           | Type   | Description                                                                                                                                                                                                                                                                 |
|------------------------------------------------------------------------------------------------------------------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>annotated_fall_transition_delta_min</code>                                                                                         | float  | This attribute shows the additional transition time added to the minimum falling transition time on a port. The additional transition time is set by either the <code>set_annotated_transition -delta_only</code> command or by PrimeTime SI during the crosstalk analysis. |
| <code>annotated_rise_transition_delta_max</code>                                                                                         | float  | This attribute shows the additional transition time added to the maximum rising transition time on a port. The additional transition time is set by either the <code>set_annotated_transition -delta_only</code> command or by PrimeTime SI during the crosstalk analysis.  |
| <code>annotated_rise_transition_delta_min</code>                                                                                         | float  | This attribute shows the additional transition time added to the minimum rising transition time on a port. The additional transition time is set by either the <code>set_annotated_transition -delta_only</code> command or by PrimeTime SI during the crosstalk analysis.  |
| <code>arrival_window</code>                                                                                                              | string | The minimum and maximum arrivals for rising and falling transitions.                                                                                                                                                                                                        |
| <code>cached_c1_max_fall</code><br><code>cached_c1_max_rise</code><br><code>cached_c1_min_fall</code><br><code>cached_c1_min_rise</code> | float  | The value of C1 (first capacitance of CCS receiver model) stored during the delay calculation.                                                                                                                                                                              |
| <code>cached_c2_max_fall</code><br><code>cached_c2_max_rise</code><br><code>cached_c2_min_fall</code><br><code>cached_c2_min_rise</code> | float  | The value of C2 (first capacitance of CCS receiver model) stored during the delay calculation.                                                                                                                                                                              |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                                                                               | Type   | Description                                                                                                                                                                                                                                                                                                                                                          |
|----------------------------------------------------------------------------------------------|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cached_ceff_max_fall<br>cached_ceff_max_rise<br>cached_ceff_min_fall<br>cached_ceff_min_rise | float  | Worst effective capacitance of a net with detailed parasitics stored during the delay calculation that is connected to this input port. This requires the <code>rc_cache_min_max_rise_fall_ceff</code> variable to be set to <code>true</code> before the timing update.                                                                                             |
| case_value                                                                                   | string | The user-specified logic value of a pin or port propagated from a case analysis or logic constant in the design.                                                                                                                                                                                                                                                     |
| ceff_params_max                                                                              | string | The parameters used to find the maximum effective capacitance for each timing arc feeding a driver pin attached to an annotated RC network. The parameters specify a linear driver model ( <code>rd</code> = drive resistance, <code>t0</code> = start of voltage ramp, <code>delta_t</code> = duration of voltage ramp, <code>ceff</code> = effective capacitance). |
| ceff_params_min                                                                              | string | The parameters used to find the minimum effective capacitance for each timing arc feeding a driver pin attached to an annotated RC network. The parameters specify a linear driver model ( <code>rd</code> = drive resistance, <code>t0</code> = start of voltage ramp, <code>delta_t</code> = duration of voltage ramp, <code>ceff</code> = effective capacitance). |
| clock_latency_fall_max                                                                       | float  | The user-specified maximum fall latency (insertion delay) for clock networks through a port. Set with the <code>set_clock_latency</code> command.                                                                                                                                                                                                                    |
| clock_latency_fall_min                                                                       | float  | The user-specified minimum fall latency (insertion delay) for clock networks through a port. Set with the <code>set_clock_latency</code> command.                                                                                                                                                                                                                    |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                                   | Type  | Description                                                                                                                                       |
|--------------------------------------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>clock_latency_rise_max</code>              | float | The user-specified maximum rise latency (insertion delay) for clock networks through a port. Set with the <code>set_clock_latency</code> command. |
| <code>clock_latency_rise_min</code>              | float | The user-specified minimum rise latency (insertion delay) for clock networks through a port. Set with the <code>set_clock_latency</code> command. |
| <code>clock_source_latency_early_fall_max</code> | float | The maximum early falling source latency. Set with the <code>set_clock_latency</code> command.                                                    |
| <code>clock_source_latency_early_fall_min</code> | float | The minimum early falling source latency. Set with the <code>set_clock_latency</code> command.                                                    |
| <code>clock_source_latency_early_rise_max</code> | float | The maximum early rising source latency. Set with the <code>set_clock_latency</code> command.                                                     |
| <code>clock_source_latency_early_rise_min</code> | float | The minimum early rising source latency. Set with the <code>set_clock_latency</code> command.                                                     |
| <code>clock_source_latency_late_fall_max</code>  | float | The maximum late falling source latency. Set with the <code>set_clock_latency</code> command.                                                     |
| <code>clock_source_latency_late_fall_min</code>  | float | The minimum late falling source latency. Set with the <code>set_clock_latency</code> command.                                                     |
| <code>clock_source_latency_late_rise_max</code>  | float | The maximum late rising source latency. Set with the <code>set_clock_latency</code> command.                                                      |
| <code>clock_source_latency_late_rise_min</code>  | float | The minimum late rising source latency. Set with the <code>set_clock_latency</code> command.                                                      |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                           | Type                    | Description                                                                                                                                                                                     |
|------------------------------------------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>clocks</code>                      | <code>collection</code> | The collection of clock objects which propagate through this port. It is undefined if no clocks are present.                                                                                    |
| <code>connection_class</code>            | <code>string</code>     | Specifies the connection class string for a port.                                                                                                                                               |
| <code>constant_value</code>              | <code>string</code>     | The logic value of a port tied to logic constant zero or one in the netlist.                                                                                                                    |
| <code>constraining_max_transition</code> | <code>float</code>      | Returns the most constraining user-defined maximum transition value at a port.                                                                                                                  |
| <code>direction</code>                   | <code>string</code>     | The direction of a port. Value can be in, out, inout, or internal. The <code>port_direction</code> attribute is a synonym for <code>direction</code> . You cannot set this attribute.           |
| <code>disable_timing</code>              | <code>Boolean</code>    | This attribute is <code>true</code> if the timing for the port has been marked as disabled with the <code>set_disable_timing</code> command.                                                    |
| <code>drive_resistance_fall_max</code>   | <code>float</code>      | A floating-point value representing the linear drive resistance for falling delays and maximum conditions, associated with an input or inout port. Set with the <code>set_drive</code> command. |
| <code>drive_resistance_fall_min</code>   | <code>float</code>      | A floating-point value representing the linear drive resistance for falling delays and minimum conditions, associated with an input or inout port. Set with the <code>set_drive</code> command. |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                             | Type       | Description                                                                                                                                                                                    |
|--------------------------------------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>drive_resistance_rise_max</code>     | float      | A floating-point value representing the linear drive resistance for rising delays and maximum conditions, associated with an input or inout port. Set with the <code>set_drive</code> command. |
| <code>drive_resistance_rise_min</code>     | float      | A floating-point value representing the linear drive resistance for rising delays and minimum conditions, associated with an input or inout port. Set with the <code>set_drive</code> command. |
| <code>driver_model_scaling_libs_max</code> | collection | The collection of library objects used for driver model scaling, where applicable, for libs maximum analysis.                                                                                  |
| <code>driver_model_scaling_libs_min</code> | collection | The collection of library objects used for driver model scaling, where applicable, for libs minimum analysis.                                                                                  |
| <code>driver_model_type_max_fall</code>    | string     | The driver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for maximum fall analysis.                                                                                            |
| <code>driver_model_type_max_rise</code>    | string     | The driver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for maximum rise analysis.                                                                                            |
| <code>driver_model_type_min_fall</code>    | string     | The driver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for minimum fall analysis.                                                                                            |
| <code>driver_model_type_min_rise</code>    | string     | The driver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for minimum rise analysis.                                                                                            |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                              | Type    | Description                                                                                                                                                                                                                         |
|---------------------------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>driving_cell_dont_scale</code>        | Boolean | When <code>true</code> , indicates not to scale the transition time on the port using the driving cell. Otherwise the transition time is scaled by operating condition factors. Set with the <code>set_driving_cell</code> command. |
| <code>driving_cell_fall_max</code>          | string  | A string that names a library cell from which to copy maximum fall drive capability to be used in fall transition calculation for the port. Set with the <code>set_driving_cell</code> command.                                     |
| <code>driving_cell_fall_min</code>          | string  | A string that names a library cell from which to copy the minimum fall drive capability to be used in fall transition calculation for the port. Set with the <code>set_driving_cell</code> command.                                 |
| <code>driving_cell_from_pin_fall_max</code> | string  | A string that names the <code>driving_cell_fall_max</code> input pin to be used to find timing arc maximum fall drive capability. Set with the <code>set_driving_cell</code> command.                                               |
| <code>driving_cell_from_pin_fall_min</code> | string  | A string that names the <code>driving_cell_fall_min</code> input pin to be used to find timing arc minimum fall drive capability. Set with the <code>set_driving_cell</code> command.                                               |
| <code>driving_cell_from_pin_rise_max</code> | string  | A string that names the <code>driving_cell_rise_max</code> input pin to be used to find timing arc rise drive capability. Set with the <code>set_driving_cell</code> command.                                                       |
| <code>driving_cell_from_pin_rise_min</code> | string  | A string that names the <code>driving_cell_rise_min</code> input pin to be used to find timing arc rise drive capability. Set with the <code>set_driving_cell</code> command.                                                       |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                                 | Type   | Description                                                                                                                                                                                                                                                                                                                               |
|------------------------------------------------|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>driving_cell_library_fall_max</code>     | string | A string that names the library in which to find the <code>driving_cell_fall_max</code> . Set with the <code>set_driving_cell</code> command.                                                                                                                                                                                             |
| <code>driving_cell_library_fall_min</code>     | string | A string that names the library in which to find the <code>driving_cell_fall_min</code> . Set with the <code>set_driving_cell</code> command.                                                                                                                                                                                             |
| <code>driving_cell_library_rise_max</code>     | string | A string that names the library in which to find the <code>driving_cell_rise_max</code> . Set with the <code>set_driving_cell</code> command.                                                                                                                                                                                             |
| <code>driving_cell_library_rise_min</code>     | string | A string that names the library in which to find the <code>driving_cell_rise_min</code> . Set with the <code>set_driving_cell</code> command.                                                                                                                                                                                             |
| <code>driving_cell_max_fall_itrans_fall</code> | float  | The value of the maximum input transition for the driving cell that was associated with a port by the <code>set_driving_cell</code> command. This attribute represents the falling transition at the <code>from_pin</code> of the driving cell that is used to compute the falling transition value at a pin that drives the port.        |
| <code>driving_cell_max_fall_itrans_rise</code> | float  | The value of the maximum input transition for the driving cell that was associated with a port by the <code>set_driving_cell</code> command. This attribute represents the falling transition value at the <code>from_pin</code> of the driving cell that is used to compute the rising transition value at the pin that drives the port. |



*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                                 | Type  | Description                                                                                                                                                                                                                                                                                                                                        |
|------------------------------------------------|-------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>driving_cell_max_rise_itrans_fall</code> | float | The value of the maximum input transition for the driving cell that was associated with a port by the <code>set_driving_cell</code> command. This attribute represents the rising transition value at the <code>from_pin</code> of the driving cell that is used to compute falling transition value at the pin that drives the port.              |
| <code>driving_cell_max_rise_itrans_rise</code> | float | The value of the maximum input transition for the driving cell that was associated with a port by the <code>set_driving_cell</code> command. This attribute represents the rising transition value at the <code>from_pin</code> of the driving cell that is used to compute the rising transition value at the pin that drives the port.           |
| <code>driving_cell_min_fall_itrans_fall</code> | float | The value of the minimum input transition for the driving cell that was associated with a port by the <code>set_driving_cell</code> command. This attribute represents the minimum falling transition value at the <code>from_pin</code> of the driving cell that is used to compute the falling transition value at the pin that drives the port. |
| <code>driving_cell_min_fall_itrans_rise</code> | float | The value of the minimum input transition for the driving cell that was associated with a port by the <code>set_driving_cell</code> command. This attribute represents the minimum falling transition value at the <code>from_pin</code> of the driving cell that is used to compute the rising transition value at the pin that drives the port.  |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                                 | Type    | Description                                                                                                                                                                                                                                                                                                                                     |
|------------------------------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>driving_cell_min_rise_itrans_fall</code> | float   | The value of the minimum input transition for the driving cell that was associated with a port by the <code>set_driving_cell</code> command. This attribute represents the minimum rise transition value at the <code>from_pin</code> of the driving cell that is used to compute the falling transition value at the pin that drives the port. |
| <code>driving_cell_min_rise_itrans_rise</code> | float   | The value of the minimum input transition for the driving cell that was associated with a port by the <code>set_driving_cell</code> command. This attribute represents the minimum rise transition value at the <code>from_pin</code> of the driving cell that is used to compute the rising transition value at the pin that drives the port.  |
| <code>driving_cell_multiply_by</code>          | float   | A floating-point value that multiplies the transition time of the port marked with this attribute. Set with the <code>set_driving_cell</code> command.                                                                                                                                                                                          |
| <code>driving_cell_no_design_rule</code>       | Boolean | This attribute is <code>true</code> if driving cell information has been set on a port with <code>set_driving_cell - no_design_rule</code> . If <code>true</code> , the driving cell's design rule limits ( <code>max_capacitance</code> and so forth) are not used for the port.                                                               |
| <code>driving_cell_pin_fall_max</code>         | string  | A string that names the <code>driving_cell_fall_max</code> output pin to be used to find timing arc fall drive capability. Set with the <code>set_driving_cell</code> command.                                                                                                                                                                  |
| <code>driving_cell_pin_fall_min</code>         | string  | A string that names the <code>driving_cell_fall_min</code> output pin to be used to find timing arc fall drive capability. Set with the <code>set_driving_cell</code> command.                                                                                                                                                                  |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                         | Type   | Description                                                                                                                                                                                         |
|----------------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>driving_cell_pin_rise_max</code> | string | A string that names the <code>driving_cell_rise_max</code> output pin to be used to find timing arc rise drive capability. Set with the <code>set_driving_cell</code> command.                      |
| <code>driving_cell_pin_rise_min</code> | string | A string that names the <code>driving_cell_rise_min</code> output pin to be used to find timing arc rise drive capability. Set with the <code>set_driving_cell</code> command.                      |
| <code>driving_cell_rise_max</code>     | string | A string that names a library cell from which to copy maximum rise drive capability to be used in rise transition calculation for the port. Set with the <code>set_driving_cell</code> command.     |
| <code>driving_cell_rise_min</code>     | string | A string that names a library cell from which to copy the minimum rise drive capability to be used in rise transition calculation for the port. Set with the <code>set_driving_cell</code> command. |
| <code>effective_capacitance_max</code> | float  | The effective capacitance for maximum operating conditions. Valid only for driver pins attached to annotated RC networks.                                                                           |
| <code>effective_capacitance_min</code> | float  | The effective capacitance for minimum operating conditions. Valid only for driver pins attached to annotated RC networks.                                                                           |
| <code>escaped_full_name</code>         | string | Contains the name of the cell. Any literal hierarchy characters are escaped with a backslash.                                                                                                       |
| <code>fanout_load</code>               | float  | Specifies the fanout load on output ports. Set with the <code>set_fanout_load</code> command.                                                                                                       |
| <code>full_name</code>                 | string | The name of a port. You cannot set this attribute.                                                                                                                                                  |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                                                                                                   | Type  | Description                                                                                                                                                                                             |
|------------------------------------------------------------------------------------------------------------------|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hold_uncertainty                                                                                                 | float | A floating-point value that specifies the clock uncertainty (skew) of a clock used for hold (and other minimum delay) timing checks. Set with the <code>set_clock_uncertainty</code> command.           |
| ideal_latency_max_fall<br>ideal_latency_max_rise<br>ideal_latency_min_fall<br>ideal_latency_min_rise             | float | Ideal delay value annotated on a port in an ideal network, using the <code>set_ideal_latency</code> command.                                                                                            |
| ideal_transition_max_fall<br>ideal_transition_max_rise<br>ideal_transition_min_fall<br>ideal_transition_min_rise | float | Ideal transition time annotated on a port in an ideal network, using the <code>set_ideal_transition</code> command.                                                                                     |
| input_transition_fall_max                                                                                        | float | A floating-point value representing the fixed transition time for falling delays, maximum conditions associated with an input or inout port. Set with the <code>set_input_transition</code> command.    |
| input_transition_fall_min                                                                                        | float | A floating-point value representing the fixed transition time for falling delays and minimum conditions associated with an input or inout port. Set with the <code>set_input_transition</code> command. |
| input_transition_rise_max                                                                                        | float | A floating-point value representing the fixed transition time for rising delays and maximum conditions associated with an input or inout port. Set with the <code>set_input_transition</code> command.  |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                                                            | Type    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>input_transition_rise_min</code>                                    | float   | A floating-point value representing the fixed transition time for rising delays and minimum conditions associated with an input or inout port. Set with the <code>set_input_transition</code> command.                                                                                                                                                                                                                                                                                                         |
| <code>is_clock_used_as_clock</code><br><code>is_clock_used_as_data</code> | Boolean | When both <code>true</code> , the clock through the specified pin acts as both clock and data. When <code>is_clock_used_as_clock</code> is <code>true</code> and <code>is_clock_used_as_data</code> is <code>false</code> , the clock through the specified pin acts only as a clock and never as data. When <code>is_clock_used_as_data</code> is <code>true</code> and <code>is_clock_used_as_clock</code> is <code>false</code> , the clock through the specified pin acts only as data and never as clock. |
| <code>is_driver_scaled_max_fall</code>                                    | Boolean | Specifies whether the port uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for maximum fall analysis.                                                                                                                                                                                                                                                                                                                                                    |
| <code>is_driver_scaled_max_rise</code>                                    | Boolean | Specifies whether the port uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for maximum rise analysis.                                                                                                                                                                                                                                                                                                                                                    |
| <code>is_driver_scaled_min_fall</code>                                    | Boolean | Specifies whether the port uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> for minimum fall analysis.                                                                                                                                                                                                                                                                                                                                                      |
| <code>is_driver_scaled_min_rise</code>                                    | Boolean | Specifies whether the port uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for minimum rise analysis.                                                                                                                                                                                                                                                                                                                                                    |
| <code>is_ideal</code>                                                     | Boolean | This attribute is <code>true</code> if the port has been marked ideal using the <code>set_ideal_network</code> command.                                                                                                                                                                                                                                                                                                                                                                                        |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                           | Type    | Description                                                                                                                                                                                                                                                                |
|------------------------------------------|---------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>is_receiver_scaled_max_fall</code> | Boolean | Specifies whether the port uses scaling between libraries for the receiver model, either <code>true</code> or <code>false</code> , for maximum fall analysis.                                                                                                              |
| <code>is_receiver_scaled_max_rise</code> | Boolean | Specifies whether the port uses scaling between libraries for the receiver model, either <code>true</code> or <code>false</code> , for maximum rise analysis.                                                                                                              |
| <code>is_receiver_scaled_min_fall</code> | Boolean | Specifies whether the port uses scaling between libraries for the receiver model, either <code>true</code> or <code>false</code> for minimum fall analysis.                                                                                                                |
| <code>is_receiver_scaled_min_rise</code> | Boolean | Specifies whether the port uses scaling between libraries for the receiver model, either <code>true</code> or <code>false</code> , for minimum rise analysis.                                                                                                              |
| <code>max_capacitance</code>             | float   | A floating-point number that sets the maximum capacitance value for input, output, or bidirectional ports, and designs. The units must be consistent with those of the technology library used during optimization. Set with the <code>set_max_capacitance</code> command. |
| <code>max_fall_delay</code>              | float   | A floating-point value that specifies the maximum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_max_delay</code> command.                                                                                          |
| <code>max_fall_slack</code>              | float   | A floating-point value representing the worst slack at a port for falling maximum path delays. This attribute is valid for path endpoints (register data pins and primary outputs) after timing has been updated. You cannot set this attribute.                           |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name               | Type  | Description                                                                                                                                                                                                                               |
|------------------------------|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>max_fanout</code>      | float | Specifies the maximum fanout load for the net connected to this port. PrimeTime ensures that the fanout load on this port is less than the specified value. Set with the <code>set_max_fanout</code> command                              |
| <code>max_rise_delay</code>  | float | A floating-point value that specifies the maximum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_max_delay</code> command.                                                          |
| <code>max_rise_slack</code>  | float | A floating-point value representing the worst slack at a port for rising maximum path delays. This attribute is valid for path endpoints (register data pins and primary outputs) after timing is updated. You cannot set this attribute. |
| <code>max_transition</code>  | float | Specifies the maximum transition time for the net connected to this port. The compile command ensures that value. Set with the <code>set_max_transition</code> command.                                                                   |
| <code>min_capacitance</code> | float | A floating-point number that sets the minimum capacitance value for input and bidirectional ports. The units must be consistent with those of the technology library used. Set with the <code>set_min_capacitance</code> command.         |
| <code>min_fall_delay</code>  | float | A floating-point value that specifies the minimum falling delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_min_delay</code> command.                                                         |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                        | Type   | Description                                                                                                                                                                                                                                |
|---------------------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>min_fall_slack</code>           | float  | A floating-point value representing the worst slack at a port for falling minimum path delays. This attribute is valid for path endpoints (register data pins and primary outputs) after timing is updated. You cannot set this attribute. |
| <code>min_fanout</code>               | float  | A floating-point value representing the minimum fanout design rule limit for a port. Set with the <code>set_min_fanout</code> command.                                                                                                     |
| <code>min_rise_delay</code>           | float  | A floating-point value that specifies the minimum rising delay on ports, clocks, pins, cells, or on paths between such objects. Set with the <code>set_min_delay</code> command.                                                           |
| <code>min_rise_slack</code>           | float  | A floating-point value representing the worst slack at a port for rising minimum path delays. This attribute is valid for path endpoints (register data pins and primary outputs) after timing is updated. You cannot set this attribute.  |
| <code>min_transition</code>           | float  | A floating-point value representing the minimum transition time design rule limit for a port. Set with the <code>set_min_transition</code> command.                                                                                        |
| <code>object_class</code>             | string | The class of the object. This is a constant, equal to port. You cannot set this attribute.                                                                                                                                                 |
| <code>pin_capacitance_max</code>      | float  | A floating-point value representing the pin capacitance of a port for maximum conditions (wire capacitance is not included). Set with the <code>set_load</code> command.                                                                   |
| <code>pin_capacitance_max_fall</code> | float  | Returns the maximum fall capacitance of the port. This attribute is read-only; you cannot change the setting.                                                                                                                              |



*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                        | Type   | Description                                                                                                                                                                                                     |
|---------------------------------------|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>pin_capacitance_max_rise</code> | float  | Returns the maximum rise capacitance of the port. This attribute is read-only; you cannot change the setting.                                                                                                   |
| <code>pin_capacitance_min</code>      | float  | A floating-point value representing the pin capacitance of a port for minimum conditions (wire capacitance is not included). Set with the <code>set_load</code> command.                                        |
| <code>pin_capacitance_min_fall</code> | float  | Returns the minimum fall capacitance of the port. This attribute is read-only; you cannot change the setting.                                                                                                   |
| <code>pin_capacitance_min_rise</code> | float  | Returns the minimum rise capacitance of the port. This attribute is read-only; you cannot change the setting.                                                                                                   |
| <code>port_direction</code>           | string | The direction of a port. Value can be in, out, or inout. This attribute exists for backward compatibility with <code>dc_shell</code> . See the <code>direction</code> attribute. You cannot set this attribute. |
| <code>power_base_clock</code>         | string | The name of the base clock associated with this port. If the port belongs to multiple clock domains, the <code>power_base_clock</code> attribute is set to the fastest of the clocks.                           |
| <code>power_rail_voltage_max</code>   | float  | Voltage value on port or pin object for maximum condition                                                                                                                                                       |
| <code>power_rail_voltage_min</code>   | float  | Voltage value on port or pin object for minimum condition                                                                                                                                                       |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                               | Type       | Description                                                                                                                                                                                                                                                                                     |
|----------------------------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>propagated_clock</code>                | Boolean    | Specifies that the clock edge times be delayed by propagating the values through the clock network. Affects all sequential cells in the transitive fanout of this port. If this attribute is not present, PrimeTime assumes ideal clocking. Set with <code>set_propagated_clock</code> command. |
| <code>receiver_model_scaling_libs_max</code> | collection | The collection of library objects used for receiver model scaling, where applicable, for libs maximum analysis.                                                                                                                                                                                 |
| <code>receiver_model_scaling_libs_min</code> | collection | The collection of library objects used for receiver model scaling, where applicable, for libs minimum analysis.                                                                                                                                                                                 |
| <code>receiver_model_type_max_fall</code>    | string     | The receiver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for type maximum fall analysis.                                                                                                                                                                                      |
| <code>receiver_model_type_max_rise</code>    | string     | The receiver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for type maximum rise analysis.                                                                                                                                                                                      |
| <code>receiver_model_type_min_fall</code>    | string     | The receiver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for type minimum fall analysis.                                                                                                                                                                                      |
| <code>receiver_model_type_min_rise</code>    | string     | The receiver model type, either “basic” (NLDM) or “advanced” (CCS Timing), for type minimum rise analysis.                                                                                                                                                                                      |
| <code>setup_uncertainty</code>               | float      | A floating-point value that specifies the clock uncertainty (skew) of a clock used for setup (and other maximum delay) timing checks. Set with the <code>set_clock_uncertainty</code> command.                                                                                                  |

**Table 12-12** *Attributes of the port Object Class (Continued)*

| Attribute name                                                                                                                                                 | Type       | Description                                                                                                                                                                                                                                                                                                                                                                                                                             |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| si_noise_active_aggressors_above_high<br>si_noise_active_aggressors_above_low<br>si_noise_active_aggressors_below_high<br>si_noise_active_aggressors_below_low | collection | This attribute returns a collection of active aggressor nets for the input port, considering crosstalk noise bumps in the above-high, below-high, above-low, or below-low region. An active aggressor is an aggressor that contributes to the worst-case noise bump on the victim net.                                                                                                                                                  |
| si_noise_bumps_above_high<br>si_noise_bumps_above_low<br>si_noise_bumps_below_high<br>si_noise_bumps_below_low                                                 | string     | This attribute returns a string that lists the aggressor nets for the input port and their corresponding coupled bump heights and widths, considering noise bumps in the above-high, below-high, above-low, or below-low region. The format of the string is:<br><pre>           {{aggr1_name height width}            {aggr2_name height width}            ... }         </pre> Height is in volts and width is in library time units. |
| si_noise_height_factor_above_high<br>si_noise_height_factor_above_low<br>si_noise_height_factor_below_high<br>si_noise_height_factor_below_low                 | float      | This attribute returns the noise height derating factor for the specified ports, in the above-high, below-high, above-low, or below-low region.                                                                                                                                                                                                                                                                                         |
| si_noise_height_offset_above_high<br>si_noise_height_offset_above_low<br>si_noise_height_offset_below_high<br>si_noise_height_offset_below_low                 | float      | This attribute returns the noise height offset derating factor for the specified ports, in the above-high, below-high, above-low, or below-low region.                                                                                                                                                                                                                                                                                  |
| si_noise_prop_bumps_above_high<br>si_noise_prop_bumps_above_low<br>si_noise_prop_bumps_below_high<br>si_noise_prop_bumps_below_low                             | string     | This attribute returns a string that shows the bump height and width at the input port caused by noise propagation, in the above-high, below-high, above-low, or below-low region. The format of the string is:<br><pre>           {height width}         </pre> Height is in volts and width is in library time units.                                                                                                                 |

**Table 12-12** *Attributes of the port Object Class (Continued)*

| Attribute name                                                                                                                                     | Type       | Description                                                                                                                                                                                                                                                                                                     |
|----------------------------------------------------------------------------------------------------------------------------------------------------|------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| si_noise_slack_above_high<br>si_noise_slack_above_low<br>si_noise_slack_below_high<br>si_noise_slack_below_low                                     | float      | This attribute returns the amount of noise slack for the port in the above-high, below-high, above-low, or below-low region.                                                                                                                                                                                    |
| si_noise_total_bump_above_high<br>si_noise_total_bump_above_low<br>si_noise_total_bump_below_high<br>si_noise_total_bump_below_low                 | string     | This attribute returns a string that shows the total bump height and width at the input port caused by crosstalk and noise propagation, in the above-high, below-high, above-low, or below-low region. The format of the string is:<br>{height width}<br>Height is in volts and width is in library time units. |
| si_noise_width_factor_above_high<br>si_noise_width_factor_above_low<br>si_noise_width_factor_below_high<br>si_noise_width_factor_below_low         | float      | This attribute returns the noise width derating factor for the specified ports, in the above-high, below-high, above-low, or below-low region.                                                                                                                                                                  |
| si_noise_worst_prop_arc_above_high<br>si_noise_worst_prop_arc_above_low<br>si_noise_worst_prop_arc_below_high<br>si_noise_worst_prop_arc_below_low | collection | The cell arc that corresponds to the worst noise propagation to an input port with a driving cell.                                                                                                                                                                                                              |
| temperature_max                                                                                                                                    | float      | Maximum temperature specified for the cell through the operating condition specification or application of the <code>set_temperature</code> command.                                                                                                                                                            |
| temperature_min                                                                                                                                    | float      | Minimum temperature specified for the cell through the operating condition specification or application of the <code>set_temperature</code> command.                                                                                                                                                            |
| user_case_value                                                                                                                                    | string     | The user-specified logic value of a pin or port.                                                                                                                                                                                                                                                                |

*Table 12-12 Attributes of the port Object Class (Continued)*

| Attribute name                               | Type   | Description                                                                                                                                                                                       |
|----------------------------------------------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wire_capacitance_max<br>wire_capacitance_min | float  | A floating-point value representing the wire capacitance of a port for maximum/minimum conditions (pin capacitance is not included). Set with the <code>set_load</code> command.                  |
| wire_capacitance_max_fall                    | float  | Returns the maximum fall wire capacitance of the net. This attribute is read-only; you cannot change the setting.                                                                                 |
| wire_capacitance_max_rise                    | float  | Returns the maximum rise wire capacitance of the net. This attribute is read-only; you cannot change the setting.                                                                                 |
| wire_capacitance_min_fall                    | float  | Returns the minimum fall wire capacitance of the net. This attribute is read-only; you cannot change the setting.                                                                                 |
| wire_capacitance_min_rise                    | float  | Returns the minimum rise wire capacitance of the net. This attribute is read-only; you cannot change the setting.                                                                                 |
| wire_load_model_max                          | string | The name of a wire load model (for maximum conditions) that can be used for prelayout wire load estimation of the net connected to a port. Set with the <code>set_wire_load_model</code> command. |
| wire_load_model_min                          | string | The name of a wire load model (for minimum conditions) that can be used for prelayout wire load estimation of the net connected to a port. Set with the <code>set_wire_load_model</code> command. |

## Timing Arc Object Class Attributes

Table 12-13 lists the timing arc object class attributes.

*Table 12-13 Attributes of the timing\_arc Object Class*

| Attribute name                              | Type  | Description                                                                                                                                                                                                                    |
|---------------------------------------------|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>annotated_delay_delta_max_fall</code> | float | Specifies a delay that is added to the maximum falling delay of a timing arc. The additional delay is set by either the <code>set_annotated_delay -delta_only</code> command or by PrimeTime SI during the crosstalk analysis. |
| <code>annotated_delay_delta_max_rise</code> | float | Specifies a delay that is added to the maximum rising delay of a timing arc. The additional delay is set by either the <code>set_annotated_delay -delta_only</code> command or by PrimeTime SI during the crosstalk analysis.  |
| <code>annotated_delay_delta_min_fall</code> | float | Specifies a delay that is added to the minimum falling delay of a timing arc. The additional delay is set by either the <code>set_annotated_delay -delta_only</code> command or by PrimeTime SI during the crosstalk analysis. |
| <code>annotated_delay_delta_min_rise</code> | float | Specifies a delay that is added to the minimum rising delay of a timing arc. The additional delay is set by either the <code>set_annotated_delay -delta_only</code> command or by PrimeTime SI during the crosstalk analysis.  |
| <code>delay_max_fall</code>                 | float | A floating-point value representing the maximum falling delay of the timing arc.                                                                                                                                               |
| <code>delay_max_rise</code>                 | float | A floating-point value representing the maximum rising delay of the timing arc.                                                                                                                                                |
| <code>delay_min_fall</code>                 | float | A floating-point value representing the minimum falling delay of the timing arc.                                                                                                                                               |
| <code>delay_min_rise</code>                 | float | A floating-point value representing the minimum rising delay of the timing arc.                                                                                                                                                |

*Table 12-13 Attributes of the timing\_arc Object Class (Continued)*

| Attribute name                             | Type       | Description                                                                                                                                                                   |
|--------------------------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>from_pin</code>                      | collection | A collection containing the from pin of the timing arc.                                                                                                                       |
| <code>is_annotated_fall_max</code>         | Boolean    | Returns <code>true</code> if the maximum falling delay of the timing arc is back-annotated.                                                                                   |
| <code>is_annotated_fall_min</code>         | Boolean    | Returns <code>true</code> if the minimum falling delay of the timing arc is back-annotated.                                                                                   |
| <code>is_annotated_rise_max</code>         | Boolean    | Returns <code>true</code> if the maximum rising delay of the timing arc is back-annotated.                                                                                    |
| <code>is_annotated_rise_min</code>         | Boolean    | Returns <code>true</code> if the minimum rising delay of the timing arc is back-annotated.                                                                                    |
| <code>is_cellarc</code>                    | Boolean    | Returns <code>true</code> if the timing arc is a cell arc, and <code>false</code> for net arcs.                                                                               |
| <code>is_constraint_scaled_max_fall</code> | Boolean    | Specifies whether the constraint timing arcs uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for maximum fall analysis. |
| <code>is_constraint_scaled_max_rise</code> | Boolean    | Specifies whether the constraint timing arcs uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for maximum rise analysis. |
| <code>is_constraint_scaled_min_fall</code> | Boolean    | Specifies whether the constraint timing arcs uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for minimum fall analysis. |
| <code>is_constraint_scaled_min_rise</code> | Boolean    | Specifies whether the constraint timing arcs uses scaling between libraries for the driver model, either <code>true</code> or <code>false</code> , for minimum rise analysis. |

*Table 12-13 Attributes of the timing\_arc Object Class (Continued)*

| Attribute name           | Type       | Description                                                                                                                                                                                                                                                                                                                                                                                        |
|--------------------------|------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| is_db_inherited_disabled | Boolean    | When this attribute is <code>true</code> , it indicates that an arc is a database-inherited disabled arc. Such an arc has been disabled for loop breaking by upstream tools, and PrimeTime also disables these arcs to be compliant with these tools. Set the <code>timing_keep_loop_breaking_disabled_arcs</code> variable to <code>true</code> to enable inheriting of .db disabled timing arcs. |
| is_disabled              | Boolean    | Returns <code>true</code> if the timing arc is disabled.                                                                                                                                                                                                                                                                                                                                           |
| is_user_disabled         | Boolean    | Returns <code>true</code> if the timing arc is disabled by using the <code>set_disable_timing</code> command.                                                                                                                                                                                                                                                                                      |
| mode                     | string     | Returns the mode string of the timing arc.                                                                                                                                                                                                                                                                                                                                                         |
| object_class             | string     | The class of the object. This is a constant equal to "timing_arc."                                                                                                                                                                                                                                                                                                                                 |
| sdf_cond                 | string     | Returns the SDF condition of the timing arc.                                                                                                                                                                                                                                                                                                                                                       |
| sdf_cond_end             | string     | Returns the SDF condition at the endpoint of the timing arc. Variable <code>sdf_enable_cond_start_end</code> must be <code>true</code> .                                                                                                                                                                                                                                                           |
| sdf_cond_start           | string     | Returns the SDF condition at the startpoint of the timing arc. Variable <code>sdf_enable_cond_start_end</code> must be <code>true</code> .                                                                                                                                                                                                                                                         |
| sense                    | string     | Returns the sense of the timing arc.                                                                                                                                                                                                                                                                                                                                                               |
| to_pin                   | collection | A collection containing the to-pin of the timing arc.                                                                                                                                                                                                                                                                                                                                              |
| when                     | string     | Returns the when string of the timing arc.                                                                                                                                                                                                                                                                                                                                                         |



## Timing Path Object Class Attributes

Table 12-14 lists the timing path object class attributes.

Table 12-14 Attributes of the *timing\_path* Object Class

| Attribute name                     | Type                    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------------------------|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>arrival</code>               | <code>float</code>      | The arrival time at the endpoint of the <code>timing_path</code> .                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>blocks</code>                | <code>string</code>     | A built-in block-aware attribute that you can use for dynamic category rules. This attribute is an ordered block level path; includes start, through, and end blocks.                                                                                                                                                                                                                                                                                        |
| <code>capture_clock_paths</code>   | <code>collection</code> | <p>Returns timing path collections for the capture clock. The path corresponds to the output you see from <code>report_timing -path full_clock_expanded</code>.</p> <p>If the capturing registers are clocks by a regular clock, it returns only one path in the collection. If it's a generated clock, the first path in the collection is the master clock path, followed by each dependent generated clock until it reaches the register's clock pin.</p> |
| <code>clock_uncertainty</code>     | <code>float</code>      | The clock uncertainty of the <code>timing_path</code> . The uncertainty can be defined with the <code>set_clock_uncertainty</code> command.                                                                                                                                                                                                                                                                                                                  |
| <code>common_path_pessimism</code> | <code>float</code>      | The value of the clock reconvergence common path pessimism. This attribute is defined only if you are using OCV analysis and have specified the <code>-report_clock_reconvergence_pessimism</code> option to <code>get_timing_paths</code> .                                                                                                                                                                                                                 |

**Table 12-14** *Attributes of the timing\_path Object Class (Continued)*

| Attribute name                                                                   | Type       | Description                                                                                                                                                                                                                                                                                                                         |
|----------------------------------------------------------------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| crpr_common_point                                                                | collection | Returns the clock pin corresponding to the common pin used for CRP calculation for the path.<br><br>If the launch or capture clock is ideal, the clock source is returned as the common point. If the launch clock is different than the capture clock, the attribute does not exist on the path.                                   |
| depth_cell_capture<br>depth_cell_launch<br>depth_net_capture<br>depth_net_launch | float      | Advanced on-chip variation (AOCV) calculated depth for a path.                                                                                                                                                                                                                                                                      |
| distance_cell<br>distance_net                                                    | float      | AOCV calculated distance for a path.                                                                                                                                                                                                                                                                                                |
| dominant_exception                                                               | string     | Exists only if the path has at least one timing exception. If so, it specifies the type of the dominant exception for the path, if any. The values are <code>false_path</code> , <code>multicycle_path</code> , and <code>min_max_delay</code> . See the <code>-exceptions</code> option of the <code>report_timing</code> command. |
| end_block                                                                        | string     | A built-in block-aware attribute that you can use for dynamic category rules. This attribute includes the end block.                                                                                                                                                                                                                |
| endpoint                                                                         | string     | The timing endpoint name of the <code>timing_path</code> , for example, <code>U1/U5/par_reg/D</code> .                                                                                                                                                                                                                              |
| endpoint_clock                                                                   | string     | The clock name of the clock at the path endpoint.                                                                                                                                                                                                                                                                                   |
| endpoint_clock_close_edge_type                                                   | string     | The type of clock edge (rise or fall) that closes (latches) the data.                                                                                                                                                                                                                                                               |
| endpoint_clock_close_edge_value                                                  | float      | The value of the closing edge of the endpoint clock.                                                                                                                                                                                                                                                                                |
| endpoint_clock_is_inverted                                                       | Boolean    | Returns <code>true</code> if the endpoint clock has been inverted.                                                                                                                                                                                                                                                                  |

*Table 12-14 Attributes of the timing\_path Object Class (Continued)*

| Attribute name                 | Type    | Description                                                                                                                                                                                                             |
|--------------------------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| endpoint_clock_is_propagated   | Boolean | Returns <code>true</code> if the endpoint clock is a propagated clock, <code>false</code> if it is an ideal clock. You can set a clock as propagated using the <code>set_propagated_clock</code> command.               |
| endpoint_clock_latency         | float   | The latency of the endpoint clock. If the clock is propagated, it is the computed latency (or delay) from the clock source to the endpoint. You can set clock latency using the <code>set_clock_latency</code> command. |
| endpoint_clock_open_edge_type  | string  | The type of clock edge (rise or fall) that opens the endpoint latch. If the endpoint is edge-triggered, the open and close edges are the same.                                                                          |
| endpoint_clock_open_edge_value | float   | The value of the opening edge of the endpoint clock.                                                                                                                                                                    |
| endpoint_clock_pin             | string  | The complete path name of the endpoint clock pin, for example, U23/U_reg/out_reg[2]/CP.                                                                                                                                 |
| endpoint_hold_time_value       | float   | The value of the register hold time at the timing endpoint. For example, for a flip-flop this would be the library hold time for the flip-flop cell.                                                                    |
| endpoint_is_level_sensitive    | Boolean | Returns <code>true</code> if the endpoint is a level-sensitive device, for example, a latch. Returns <code>false</code> if the endpoint is edge-triggered.                                                              |
| endpoint_output_delay_value    | float   | The value of the output delay of the timing endpoint. You can set the output delay value with <code>set_output_delay</code> .                                                                                           |
| endpoint_recovery_time_value   | float   | The value of the recovery time at the timing endpoint. Recovery and removal times are often defined for the asynchronous set and clear pins of registers.                                                               |

*Table 12-14 Attributes of the timing\_path Object Class (Continued)*

| Attribute name                | Type       | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| endpoint_removal_time_value   | float      | The value of the removal time at the timing endpoint. Recovery and removal times are often defined for the asynchronous set/clear pins of registers.                                                                                                                                                                                                                                                                                                        |
| endpoint_setup_time_value     | float      | The value of the setup time at the timing endpoint. Recovery and removal times are often defined for the asynchronous set and clear pins of registers.                                                                                                                                                                                                                                                                                                      |
| endpoint_unconstrained_reason | string     | Exists only if the path endpoint is unconstrained. If so, it specifies the reason why the endpoint is unconstrained. Possible reasons include <code>no_capture_clock</code> , <code>dangling_end_point</code> , and <code>fanin_of_disabled</code> .                                                                                                                                                                                                        |
| is_recalculated               | Boolean    | Returns <code>true</code> if the timing information for the path comes from path-based (recalculated) timing analysis.                                                                                                                                                                                                                                                                                                                                      |
| launch_clock_paths            | collection | <p>Returns timing path collections for the launch clock. The path corresponds to the output you see from <code>report_timing -path full_clock_expanded</code>.</p> <p>If the launching registers are clocks by a regular clock, it returns only one path in the collection. If it's a generated clock, the first path in the collection is the master clock path, followed by each dependent generated clock until it reaches the register's clock pin.</p> |
| object_class                  | string     | The class of the object. This is a constant, equal to <code>timing_path</code> . You cannot set this attribute.                                                                                                                                                                                                                                                                                                                                             |
| path_group                    | collection | The path group of the timing path.                                                                                                                                                                                                                                                                                                                                                                                                                          |

*Table 12-14 Attributes of the timing\_path Object Class (Continued)*

| Attribute name                       | Type                    | Description                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------------|-------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>path_type</code>               | <code>string</code>     | The type of timing path (maximum or minimum). A path for a setup check is <code>path_type</code> of maximum.                                                                                                                                                                                                                                             |
| <code>points</code>                  | <code>collection</code> | Returns a collection of the timing points that comprise a timing path. For example, the timing points listed in the left-hand column of a <code>report_timing</code> command correspond to this collection. A single timing path can consist of many timing points. You can iterate through each timing point using <code>foreach_in_collection</code> . |
| <code>required</code>                | <code>float</code>      | The required time value for the timing path. Corresponds to the data required time of a timing report.                                                                                                                                                                                                                                                   |
| <code>session_name</code>            | <code>string</code>     | The name of the saved session that the timing path belongs to. Applicable only in the interactive multi-scenario analysis flow.                                                                                                                                                                                                                          |
| <code>slack</code>                   | <code>float</code>      | The slack of the timing path. Negative values represent violated paths. Corresponds to the slack of a timing report.                                                                                                                                                                                                                                     |
| <code>start_blocks</code>            | <code>string</code>     | A built-in block-aware attribute that you can use for dynamic category rules. This attribute includes the start block.                                                                                                                                                                                                                                   |
| <code>start_end_blocks</code>        | <code>string</code>     | A built-in block-aware attribute that you can use for dynamic category rules. This attribute includes a pair of blocks consisting of the start and end blocks.                                                                                                                                                                                           |
| <code>start_end_blocks_sorted</code> | <code>string</code>     | A built-in block-aware attribute that you can use for dynamic category rules. This attribute includes a pair of blocks consisting of the start and end blocks sorted alphabetically.                                                                                                                                                                     |

*Table 12-14 Attributes of the timing\_path Object Class (Continued)*

| Attribute name                   | Type       | Description                                                                                                                                                                                                                                                |
|----------------------------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| startpoint                       | collection | The startpoint of the timing path. Corresponds to the startpoint in the header of a timing report.                                                                                                                                                         |
| startpoint_clock                 | collection | The startpoint clock name of the timing path.                                                                                                                                                                                                              |
| startpoint_clock_is_inverted     | Boolean    | Returns <code>true</code> if the startpoint clock is inverted.                                                                                                                                                                                             |
| startpoint_clock_is_propagated   | Boolean    | Returns <code>true</code> if the startpoint clock is a propagated clock, <code>false</code> if it is an ideal clock. You can set a clock as propagated using the <code>set_propagated_clock</code> command.                                                |
| startpoint_clock_latency         | float      | The latency of the startpoint clock. If the clock is propagated, it is the computed latency (or delay) from the clock source to the endpoint. You can set clock latency using the <code>set_clock_latency</code> command.                                  |
| startpoint_clock_open_edge_type  | string     | The type of clock edge (rise or fall) that launches the data.                                                                                                                                                                                              |
| startpoint_clock_open_edge_value | float      | The value of the opening edge of the startpoint clock.                                                                                                                                                                                                     |
| startpoint_input_delay_value     | float      | The value of the startpoint input delay.                                                                                                                                                                                                                   |
| startpoint_is_level_sensitive    | Boolean    | Returns <code>true</code> if the startpoint is a level-sensitive device, such as a latch. Returns <code>false</code> if the startpoint is edge-triggered.                                                                                                  |
| startpoint_unconstrained_reason  | string     | Exists only if the path startpoint is unconstrained. If so, it specifies the reason why the startpoint is unconstrained. Possible reasons include <code>no_launch_clock</code> , <code>dangling_start_point</code> , and <code>fanout_of_disabled</code> . |

*Table 12-14 Attributes of the timing\_path Object Class (Continued)*

| Attribute name                         | Type       | Description                                                                                                                                                                                                                                                                        |
|----------------------------------------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| through_blocks                         | string     | A built-in block-aware attribute that you can use for dynamic category rules. This attribute includes through blocks, but does not include the start or end blocks.                                                                                                                |
| time_borrowed_from_endpoint            | float      | Returns the amount of time borrowed from the timing endpoint. Time borrowing occurs in paths involving level-sensitive devices.                                                                                                                                                    |
| time_lent_to_startpoint                | float      | Returns the amount of time lent to the timing startpoint. Time borrowing occurs in paths involving level-sensitive devices.                                                                                                                                                        |
| transparent_latch_paths                | collection | For paths with a transparent latch D-pin startpoint, if the startpoint is borrowing, the attribute returns the chain of "upstream" borrowing paths that lead up to the borrowing startpoint. To use the attribute, you must gather the path using the - trace_latch_borrow option. |
| variation_arrival                      | collection | Arrival time variation of the path.                                                                                                                                                                                                                                                |
| variation_common_path_pessimism        | collection | The variation of the clock reconvergence common path pessimism.                                                                                                                                                                                                                    |
| variation_endpoint_clock_latency       | collection | Capture-clock arrival time variation.                                                                                                                                                                                                                                              |
| variation_endpoint_hold_time_value     | collection | The variation of the register hold time at the timing endpoint.                                                                                                                                                                                                                    |
| variation_endpoint_recovery_time_value | collection | The variation of the recovery time at the timing endpoint.                                                                                                                                                                                                                         |
| variation_endpoint_removal_time_value  | collection | The variation of the removal time at the timing endpoint.                                                                                                                                                                                                                          |
| variation_endpoint_setup_time_value    | collection | The variation of the register setup time at the timing endpoint.                                                                                                                                                                                                                   |
| variation_required                     | collection | Required time variation of the path.                                                                                                                                                                                                                                               |

*Table 12-14 Attributes of the timing\_path Object Class (Continued)*

| Attribute name                     | Type       | Description                          |
|------------------------------------|------------|--------------------------------------|
| variation_slack                    | collection | Slack variation of the path.         |
| variation_startpoint_clock_latency | collection | Launch clock arrival time variation. |

## Timing Point Object Class Attributes

[Table 12-15](#) lists the point object class attributes.

*Table 12-15 Attributes of the timing\_point Object Class*

| Attribute name             | Type   | Description                                                                                                                                                                                                                                                                                                                                                        |
|----------------------------|--------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| annotated_delay_delta      | float  | Delta delay in the timing point.                                                                                                                                                                                                                                                                                                                                   |
| annotated_delta_transition | float  | Delta transition in the timing point.                                                                                                                                                                                                                                                                                                                              |
| arrival                    | string | The arrival time at the timing point, but not accounting for the following.<br>- Clock latency to the startpoint clock<br>- Time lent to the startpoint (due to latch borrowing)<br>- Input delay<br>- startpoint_clock_open_edge_value<br>These must be added to the arrival attribute value to arrive at the total arrival time based on the desired startpoint. |
| object                     | string | The object at this point in the timing path.                                                                                                                                                                                                                                                                                                                       |
| object_class               | string | The class of the object. This is a constant, equal to the <code>timing_point</code> object. You cannot set this attribute.                                                                                                                                                                                                                                         |
| rise_fall                  | string | Returns rise if the timing point is a rising-edge delay. Returns fall if the timing point is a falling-edge delay.                                                                                                                                                                                                                                                 |
| si_xtalk_bumps             | string | The crosstalk bump at the timing point.                                                                                                                                                                                                                                                                                                                            |
| slack                      | string | The slack value at the timing point.                                                                                                                                                                                                                                                                                                                               |
| transition                 | string | The transition value at the timing point.                                                                                                                                                                                                                                                                                                                          |



*Table 12-15 Attributes of the timing\_point Object Class (Continued)*

| Attribute name       | Type       | Description                                    |
|----------------------|------------|------------------------------------------------|
| variation_arrival    | collection | Arrival time variation of the timing point.    |
| variation_transition | collection | Slack time variation of the timing point.      |
| variation_transition | collection | Transition time variation of the timing point. |
| voltage              | float      | The voltage level of the pin.                  |

## Using Paths to Generate Custom Reports

Use the `get_timing_paths` command to create a collection of paths for custom reporting and other processing. You can assign these timing paths to a variable or pass them into another command. For a list of supported timing path object class attributes, see [Table 12-14 on page 12-95](#).

Each use of the `get_timing_paths` command has its own context. You cannot compare, add, or remove objects taken from different contexts. For example,

```
pt_shell> set paths1 [get_timing_paths -nworst 10]
...
pt_shell> set paths2 [get_timing_paths -nworst 100]
...
```

Even though the variables `paths1` and `paths2` contain some of the same objects, the collections cannot be compared because they come from different contexts. To do such comparisons, create one large collection containing all the objects of interest, and then perform filtering or other manipulation on that collection.

Use the `foreach_in_collection` command to iterate among the paths in the collection. The collection commands `index_collection`, `copy_collection`, `add_to_collection`, and `remove_from_collection` are not applicable to timing path collections. Use the `get_attribute` command to obtain information about the paths.

One attribute of a timing path is the `points` collection. A point corresponds to a pin or port along the path. Iterate through these points using the `foreach_in_collection` command and get the attributes on them using the `get_attribute` command. For more information, see the man pages for the `foreach_in_collection` and `get_timing_paths` commands.

## Example

You can find some detailed examples of custom timing reports in *install\_directory/auxx/pt/examples/tcl* (where *install\_directory* is the installation directory for PrimeTime).

The following procedure prints out the startpoint name, endpoint name, and the slack of the worst path in each path group:

```
proc custom_report_worst_path_per_group {} {
 echo [format "%-20s %-20s %7s" "From" "To" "Slack"]
 echo "-----"
 foreach_in_collection path [get_timing_paths] {
 set slack [get_attribute $path slack]
 set startpoint [get_attribute $path startpoint]
 set endpoint [get_attribute $path endpoint]
 echo [format "%-20s %-20s %s" [get_attribute $startpoint
full_name] \
 [get_attribute $endpoint full_name] $slack]
 }
}
```

```
pt_shell> custom_report_worst_path_per_group
```

| From   | To    | Slack  |
|--------|-------|--------|
| ffa/CP | QA    | 0.1977 |
| ffb/CP | ffd/D | 3.8834 |

This example shows worst negative slack, total negative slack, and total positive slack for the current design:

```
proc report_design_slack_information {} {
 set design_tns 0
 set design_wns 100000
 set design_tps 0
 foreach_in_collection group [get_path_groups *] {
 set group_tns 0
 set group_wns 100000
 set group_tps 0
 foreach_in_collection path [get_timing_paths -nworst 10000 \
 -group $group] {
 set slack [get_attribute $path slack]
 if {$slack < $group_wns} {
 set group_wns $slack
 if {$slack < $design_wns} {
 set design_wns $slack
 }
 }
 if {$slack < 0.0} {
 set group_tns [expr $group_tns + $slack]
 } else {
 set group_tps [expr $group_tps + $slack]
 }
 }
 }
}
```

```

 set group_tps [expr $group_tps + $slack]
 }
}
set design_tns [expr $design_tns + $group_tns]
set design_tps [expr $design_tps + $group_tps]
set group_name [get_attribute $group full_name]
echo [format "Group '%s' Worst Negative Slack : %g" $group_name
$group_wns]
echo [format "Group '%s' Total Negative Slack : %g" $group_name
$group_tns]
echo [format "Group '%s' Total Positive Slack : %g" $group_name
$group_tps]
echo ""
}
echo "-----"
echo [format "Design Worst Negative Slack : %g" $design_wns]
echo [format "Design Total Negative Slack : %g" $design_tns]
echo [format "Design Total Positive Slack : %g" $design_tps]
}

```

```
pt_shell> report_design_slack_information
```

```

Group 'CLK' Worst Negative Slack : -3.1166
Group 'CLK' Total Negative Slack : -232.986
Group 'CLK' Total Positive Slack : 4.5656

Group 'vclk' Worst Negative Slack : -4.0213
Group 'vclk' Total Negative Slack : -46.1982
Group 'vclk' Total Positive Slack : 0

Design Worst Negative Slack : -4.0213
Design Total Negative Slack : -279.184
Design Total Positive Slack : 4.5656

```

---

## Using Arcs to Generate Custom Reports

To create a collection of timing arcs for custom reporting and other processing, use the `get_timing_arcs` command. You can assign these timing arcs to a variable and get the desired attributes for further processing. For a list of supported timing arc object class attributes, see [Table 12-13 on page 12-92](#).

Use the `foreach_in_collection` command to iterate among the arcs in the collection. Use the `get_attribute` command to obtain information about the arcs. However, you cannot copy, sort, or index the collection of arcs. You can also use the `-filter` option of `get_timing_arcs` to obtain just the arcs that satisfy specified conditions, but you cannot use the `filter_collection` command to filter an already generated collection of arcs.

One attribute of a timing arc is `from_pin`, which is the pin or port from which the timing arc begins. In the same way, `to_pin` is the pin or port at which the timing arc ends. For more information about the `from_pin` and `to_pin` attributes, use the `get_attribute` command.

### Example

To list the maximum rise delay value of all the timing arcs of the cell U1 that have the `positive_unate` arc, enter

```
pt_shell> set arcs [get_timing_arcs -of_objects U1 -filter \
 "sense == positive_unate"]
_sel3
pt_shell> foreach_in_collection arc $arcs {
 echo [get_attribute $arc delay_max_rise]
 }
0.846862
0.846862
```

---

## Creating a Collection of Library Arcs

To create a collection of library arcs for custom reporting and other processing, use the `get_lib_timing_arcs` command. You can assign these library arcs to a variable and get the desired attributes for further processing.

Use the `foreach_in_collection` command to iterate among the library arcs in the collection. Use the `get_attribute` command to obtain information about the arcs. However, you cannot copy, sort, or index a collection of library arcs. You can also use the `-filter` option of `get_lib_timing_arcs` to obtain just the library arcs that satisfy specified conditions, but you cannot use the `filter_collection` command to filter a collection of library arcs. For a list of supported library timing arc object class attributes, see [Table 12-8 on page 12-40](#).

One attribute of a library timing arc is `from_lib_pin`, which is the library pin from which the timing arc begins. In the same way, `to_lib_pin` is the library pin at which the timing arc ends. To obtain information about the `from_lib_pin` and `to_lib_pin` attributes, use the `get_attribute` command. For more information, see the man page for the `foreach_in_collection` command.

### Example

To list the senses of timing arcs starting from the clock pin of a flip-flop library cell, enter

```
pt_shell> set larcs [get_lib_timing_arcs -from class/FD1/CP]
_sel5
pt_shell> foreach_in_collection larc $larcs \ {
 { echo [get_attribute $larc sense] }
 }
hold_clk_rise
setup_clk_rise
```

```
rising_edge
rising_edge
```

---

## Reporting Library Data and Driver Information

To report library data and the resulting driver model parameters for a specified library arc and conditions, you can use the `report_driver_model` command. The reported information is useful for validating library data and driver models used in delay calculation with annotated parasitics. For more information, see the `report_driver_model` man page.



# A

## Writing Mapped SDF Files

---

The Standard Delay Format (SDF) mapping feature of PrimeTime lets you specify the output format of the SDF file. You start by creating an SDF map file, which specifies the syntax and timing arcs written to the SDF file for cells in the library. For example, if you want to change the SDF output for a flip-flop in the library, you define a map for that cell. You apply the mapping by using the `-map` option of the `write_sdf` command.

The SDF mapping feature is described in the following sections:

- [Specifying Timing Labels in the Library](#)
- [Specifying the min\\_pulse\\_width Constraint](#)
- [Using SDF Mapping](#)
- [Supported SDF Mapping Functions](#)
- [SDF Mapping Assumptions](#)
- [Labeling Bus Arcs](#)

---

## Specifying Timing Labels in the Library

To reference delays of timing arcs in the SDF mapping mechanism, you need to specify a label to the timing arcs in the library. The `timing_label` attribute is written in the timing group of the library file to label a timing arc. This labels arcs for supplying values for the following SDF constructs: `IOPATH`, `SETUP`, `HOLD`, `SETUPHOLD`, `RECOVERY`, `REMOVAL`, `RECREM`, `NOCHANGE`, and `WIDTH`.

The following example shows how to label a timing arc `A_Z`.

```
cell(IV_I) {
 area : 1;
 pin(A) {
 direction : input;
 capacitance : 1;
 }
 pin(Z) {
 direction : output;
 function : "A'";
 timing() {
 related_pin : "A";
 timing_label : "A_Z";
 rise_propagation(onebyone) {
 values("0.380000");
 }
 rise_transition(tran) {
 values("0.03, 0.04, 0.05");
 }
 fall_propagation(prop) {
 values("0.15, 0.17, 0.20");
 }
 fall_transition(tran) {
 values("0.02, 0.04, 0.06");
 }
 }
 }
}
```

---

## Specifying the min\_pulse\_width Constraint

You can specify the minimum pulse width arcs in two ways in the library: by attaching attributes to the pins or by having a `min_pulse_width` group section inside the pin section. When specifying the `min_pulse_width` pin constraint, use one of the following styles when writing the library file.

### Style 1

Specify a `min_pulse_width` group within the pin group (in the library file).



- You should place the `timing_label_mpw_high` attribute in the `min_pulse_width` group to refer to the `constraint_high` attribute.
- You should place the `timing_label_mpw_low` attribute in the `min_pulse_width` group to refer to the `constraint_low` attribute.

## Style 2

Specify the constraints in the library by using the `min_pulse_width_high` and `min_pulse_width_low` attributes in the pin group.

- You should place the `timing_label_mpw_high` attribute in the pin group to refer to the `min_pulse_width_high` attribute.
- You should place the `timing_label_mpw_low` attribute in the pin group to refer to the `min_pulse_with_low` attribute.

---

## Accessing the min\_pulse\_width Constraint

To access the `min_pulse_width` values, use these functions:

- `min_rise_delay(label_high)`  
Returns the value of the `min_pulse_width_high` attribute.
- `max_rise_delay(label_high)` Same as `min_rise_delay(label_high)`.
- `min_fall_delay (label_low)`  
Returns the value of the `min_pulse_width_low` attribute.
- `max_fall_delay(label_low)` Same as `min_fall_delay(label_low)`.

For information about functions that support SDF mapping, see [Table A-2 on page A-7](#).

---

## Specifying the min\_period Constraint on a Pin

Specify the `min_period` constraint on a pin as an attribute in the pin section. Use the `min_period(pin_name)` function to take the pin name as argument and to return the value of the `min_period` constraint.

---

## Using SDF Mapping

To specify the SDF mapping file when writing out SDF for a design, use the `write_sdf` command with the `-map` option. To specify how bus bit names are written in the mapped SDF output, you can use the `-context` option. For example,

- `-context verilog`  
The bus bit names are printed using the `%s[%d]` format.
- `-context vhdl`  
The bus bit names are printed using the `%s(%d)` format.

**Note:**

Only names that are specified using the `bus()` function are affected. For more information, see [Table A-2 on page A-7](#).

To write SDF with the `-context` and `-map` option, enter

```
pt_shell> write_sdf -context verilog -map example.map example.sdf
```

**Output**

File: exmaple.sdf

```
(DELAYFILE
(SDFVERSION "OVI V2.1")
(DESIGN "ADDER")
(DATE "Tue Feb3 09:33:06 1999")
(VENDOR "nyaa")
(PROGRAM " ")
(VERSION "3.24")
(DIVIDER /)
(VOLTAGE 0:0:0)
(PROCESS "")
(TEMPERATURE 0:0:0)
(TIMESCALE 1 ns)
(CELL
(CELLTYPE "ADDER")
(INSTANCE)
(DELAY
(ABSOLUTE
(INTERCONNECT ...))
)
)
)
(CELL
(CELLTYPE "EXMP")
(INSTANCE U15)
(DELAY
(ABSOLUTE
(IOPATH A Z (1.8::4.3) (1.8::4.0))
(IOPATH IN[0] Z (3.0::1.2) (3.0::1.0))
)
)
)
(CELL
(CELLTYPE "FD02"
(INSTANCE B1/U12/DFF)
(DELAY
```

```

 (ABSOLUTE
 (IOPATH (posedge CP) Q(1.2::1.2) (1.5::1.5))
 (IOPATH (posedge CP) QN(1.2::1.2) (1.5::1.5))
 (IOPATH (negedge CD) Q (::) (2.1::2.4))
 (IOPATH (negedge CD) QN (2.1::2.4) (::))
)
 (TIMINGCHECK
 (SETUP D (posedge CP)) (2.1::2.5))
 (HOLD D (posedge CP)) (1.1::1.4))
 (WIDTH (negedge CP) (5.0))
 (RECOVERY (posedge CD) (posedge CP)) (2.5))
 (PERIOD (posedge CP) (5.0))
)
)
)
)
)

```

**Note:**

When SDF for cells are not present in the SDF mapping file, they are written with the default `write_sdf` format.

---

## SDF Mapping Notation

The notation used in presenting the syntax of the SDF mapping language follows. In the following list, *item* is a symbol for a syntax construct item.

- *item?* – Item is optional in the definition (it can appear once or not at all).
- *item\** – Item can appear zero or any number of times.
- *item+* – Item can appear one or more times (but cannot be omitted).
- **KEYWORD** – Keywords are shown in uppercase bold for easy identification but are case-insensitive.
- **VARIABLE** – Is a symbol for a variable. Variable symbols are shown in uppercase for easy identification, but are case-insensitive. Some variables are defined as one of a number of discrete choices; other variables represent user data such as names and numbers.

---

## SDF Mapping Comments

Comment lines in the SDF mapping language are in this format:

```
[white_space]* [#] [any char except new-line char]*
```

If the line has a pound sign (#) as the first non-whitespace character, it is treated as a comment line and is ignored.

**Note:**

Quoted strings can contain new-line characters in them, and can span multiple lines. Comments cannot be embedded inside a quoted string. For definitions of SDF mapping functions, see [Table A-2 on page A-7](#).

---

## SDF Mapping Variables

[Table A-1](#) lists the variables used in the SDF mapping language.

*Table A-1 SDF Mapping Variables*

| Variable          | Description                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SDFMAP_QSTRING    | A string of any legal SDF characters and spaces, including tabs and new-line characters, optionally enclosed by double quotation marks.                                                                                                                                                                                                                                 |
| SDFMAP_NUMBER     | A nonnegative (zero or positive) real number, for example, 0, 1, 3.4, .7, 0.5, 2.4e-2, 3.4e2                                                                                                                                                                                                                                                                            |
| SDFMAP_RNUMBER    | A positive, zero, or negative real number, for example, 0, 1, 0.0, -3.4, .7, -0.3, 2.4e-2, 3.4e4                                                                                                                                                                                                                                                                        |
| SDFMAP_DNUMBER    | A nonnegative integer number, for example, +12, 23, 0                                                                                                                                                                                                                                                                                                                   |
| SDFMAP_TSVALUE    | A real number followed by a unit.                                                                                                                                                                                                                                                                                                                                       |
| SDFMAP_IDENTIFIER | The name of an object. It is case sensitive. The following characters are allowed: alphanumeric characters, the underscore character (_), and the escape character (\). If you want to use a nonalphanumeric character as a part of an identifier, you must prefix it with the escape character. White spaces are not allowed with the hierarchy divider character (/). |
| SDFMAP_MAP        | A positive integer value preceded with the dollar sign (\$), for example, \$10, \$3, \$98. Used to specify a placeholder for a value in the mapping file.                                                                                                                                                                                                               |
| SDFMAP_FUNCTION   | A string that denotes the function name. Supported functions are given in <a href="#">Table A-2</a> .                                                                                                                                                                                                                                                                   |

## Supported SDF Mapping Functions

[Table A-2](#) lists the functions supported when writing mapped SDF files.

*Table A-2 SDF Mapping Functions*

| Function                        | Usage                                                    | Return value | Comment                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------------|----------------------------------------------------------|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>in</code>                 | <code>in(float)</code>                                   | float        | Natural logarithm.                                                                                                                                                                                                                                                                                                                                                                                                                       |
| <code>exp</code>                | <code>exp(float)</code>                                  | float        | Exponentiation.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <code>sqrt</code>               | <code>sqrt(float)</code>                                 | float        | Square root.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <code>max</code>                | <code>max(float, float)</code>                           | float        | Two-argument maximum function.                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>min</code>                | <code>min(float, float)</code>                           | float        | Two-argument minimum function.                                                                                                                                                                                                                                                                                                                                                                                                           |
| <code>bus</code>                | <code>bus(string)</code>                                 | string       | <p>Returns a transformed string by replacing the bus delimiter characters in the string with the appropriate bus delimiter characters as specified in the <code>write_sdf</code> command.</p> <p>The <code>-context verilog</code> option causes the string to be transformed to the <code>%s[%d]</code> format.</p> <p>The <code>-context vhdl</code> option causes the string to be transformed to the <code>%s(%d)</code> format.</p> |
| <code>pin</code>                | <code>pin(string)</code>                                 | string       | Returns the string argument passed to it.                                                                                                                                                                                                                                                                                                                                                                                                |
| <code>max_fall_delay</code>     | <code>max_fall_delay(label)</code>                       | float        | Maximum fall delay value associated with that timing label.                                                                                                                                                                                                                                                                                                                                                                              |
| <code>max_fall_delay_bus</code> | <code>max_fall_delay_bus(label, from_pin, to_pin)</code> | float        | Maximum fall delay value to be used when the timing label is associated with a bus arc.                                                                                                                                                                                                                                                                                                                                                  |

*Table A-2 SDF Mapping Functions (Continued)*

| Function                           | Usage                                                       | Return value | Comment                                                                                    |
|------------------------------------|-------------------------------------------------------------|--------------|--------------------------------------------------------------------------------------------|
| <code>max_fall_retain_delay</code> | <code>max_fall_retain_delay(label, from_pin, to_pin)</code> | float        | Maximum fall delay value to be used when the timing label is associated with a retain arc. |
| <code>max_rise_delay</code>        | <code>max_rise_delay(label)</code>                          | float        | Maximum rise delay value associated with that timing label.                                |
| <code>max_rise_delay_bus</code>    | <code>max_rise_delay_bus(label, from_pin, to_pin)</code>    | float        | Maximum rise delay value is used when the timing label is associated with a bus arc.       |
| <code>max_rise_retain_delay</code> | <code>max_rise_retain_delay(label, from_pin, to_pin)</code> | float        | Maximum rise delay value to be used when the timing label is associated with a retain arc. |
| <code>min_fall_delay</code>        | <code>min_fall_delay(label)</code>                          | float        | Minimum fall delay value associated with that timing label.                                |
| <code>min_fall_delay_bus</code>    | <code>min_fall_delay_bus(label, from_pin, to_pin)</code>    | float        | Minimum fall delay function to be used when the timing label is associated with a bus arc. |
| <code>min_fall_retain_delay</code> | <code>min_fall_retain_delay(label, from_pin, to_pin)</code> | float        | Minimum fall delay value to be used when the timing label is associated with a retain arc. |
| <code>min_period</code>            | <code>min_period(pin)</code>                                | float        | Returns the minimum period value associated with the pin name "pin" of the current cell.   |
| <code>min_rise_delay</code>        | <code>min_rise_delay(label)</code>                          | float        | Minimum rise delay value associated with that timing label.                                |
| <code>min_rise_delay_bus</code>    | <code>min_rise_delay_bus(label, from_pin, to_pin)</code>    | float        | Minimum rise delay value to be used when the timing label is associated with a bus arc.    |
| <code>min_rise_retain_delay</code> | <code>min_rise_retain_delay(label, from_pin, to_pin)</code> | float        | Minimum rise delay value to be used when the timing label is associated with a retain arc. |

## SDF Mapping File Syntax

The following example gives the complete syntax of a mapped SDF file.

```

mapping_file ::= map_header cell_map+

map_header ::= sdf_version sdf_map_name?
hierarchy_divider?
 bus_delimiter?

sdf_map_name ::= $SDF_MAP_NAME
SDFMAP_QSTRING

sdf_version ::= $SDF_VERSION SDFMAP_QSTRING
hierarchy_divider ::= $SDF_HIERARCHY_DIVIDER SDFMAP_HCHAR
bus_delimiter ::= $SDF_BUSBIT SDFMAP_QSTRING

cell_map ::= $SDF_CELL cell_name format_string
var_map ::= $SDF_CELL_END
cell_name ::= SDFMAP_QSTRING
format_string ::= SDFMAP_QSTRING

var_map ::= var_map_line+
var_map_line: ::= SDFMAP_MAP expression

expression ::= expression binary_operator expression
 ||= unary_operator expression
 ||= expression ? expression : expression
 ||= (expression)
 ||= function_call
 ||= SDFMAP_NUMBER

function_call ::= SDFMAP_FUNCTION (func_args?)
func_args ::= func_arg
 ||= func_args , func_args
func_arg ::= SDFMAP_IDENTIFIER
 ||= expression

binary_operator ::= +
 ||= -
 ||= *
 ||= /
 ||= &
 ||= |
 ||= >
 ||= <
 ||= =
binary_operator ::= !
 ||= -

```

---

## SDF Mapping Assumptions

The SDF mapping file reader assumes that the SDF format string read from the SDF mapping file has valid SDF syntax. For instance, it assumes that when the proper substitutions are made for the placeholders, the resulting SDF is syntactically (and also semantically) correct. The SDF map parser does not attempt to parse the format string to check for possible user errors. It performs the placeholder substitutions and prints the resulting string to the SDF output file.

To check the validity of the generated mapped SDF file, you can read the mapped file by using the `read_sdf` command. For example,

```
pt_shell> read_sdf -syntax_only mapped.sdf
```

---

## Bus Naming Conventions

When you specify mapping names with the `bus(string)` function, the SDF writer replaces the bus bit delimiting characters (specified by the construct `$SDF_BUSBIT QSTRING`) by using the appropriate delimiting characters.

For example, suppose the following lines are in the mapping file:

```
$SDF_BUSBIT "<>"
....
$23 bus(output_bus<5>)
```

When you specify the `write_sdf` command with the `-context verilog` option, the SDF writer prints the name as `output_bus[5]`. For more information, see [“Using SDF Mapping” on page A-3](#). If more than one set of matching bus delimiters is found in a name, the SDF writer replaces only the matching set of delimiters at the end of the name string. If you do not specify bus delimiters, the names are printed unchanged.

## Header Consistency Check for SDF Mapping Files

Each SDF mapping file defines SDF version and SDF bus delimiter characters in its header. When PrimeTime reads multiple SDF mapping files during one `write_sdf` command, it assumes all the headers have the same SDF version and bus delimiter characters.

---

## Labeling Bus Arcs

A pin in the library file can be a bus. In the following example, the timing group on the pin defines multiple arcs.



**Example 1**

```

bus(A) {
 bus_type : bus_11;
 direction : input ;
 timing() {
 related_pin : "CK" ;
 timing_type : setup_rising ;
 timing_label : "tas";
 intrinsic_rise : 1.12000 ;
 intrinsic_fall : 1.12000 ;
 }
}

```

In this example (assuming A is an 11-bit bus), 11 setup arcs are defined between bus A and CK. When you label such an arc, since only one timing label is present, PrimeTime attaches the same label (tas) to all arcs defined by the statement. When you access individual bit arcs, you should use the bus versions of the max/min rise/fall delay functions. For example, to refer to the arc corresponding to A[5], which is from CK to A[5], use the mapping functions shown in the following examples.

**Example 2**

```

max_rise_delay_bus(tas, CK, A[5]) #To get the max rise delay
min_rise_delay_bus(tas, CK, A[5]) #To get the min rise delay
max_fall_delay_bus(tas, CK, A[5]) #To get the max fall delay
min_fall_delay_bus(tas, CK, A[5]) #To get the min fall delay

```

**Example 3**

```

bus (In) {
 bus_type : "bus8";
 direction : input;
 capacitance : 1.46;
 fanout_load : 1.46;
}
bus(Q) {
 bus_type : "bus8";
 direction : output;
 timing () {
 timing_label : "In_to_Q"
 timing_sense : non_unate;
 intrinsic_rise : 2.251622;
 rise_resistance : 0.020878;
 intrinsic_fall : 2.571993;
 fall_resistance : 0.017073;
 related_pin : "In";
 }
}

```

To refer to the arc from In[6] to Q[6] and print the following SDF line:

```
(IOPATH In[6] Q[6] (1.8::4.3) (1.8::4.0))
```

The SDF mapping file format string should look like:

```
(IOPATH $1 $2 ($3::$4) ($5::$6))
```

The resulting SDF function mapping is as follows:

```
$1 bus(I1[6])
$2 bus(Q[6])
$3 min_rise_delay_bus(In_to_Q, In[6], Q[6])
$4 max_rise_delay_bus(In_to_Q, In[6], Q[6])
$5 min_fall_delay_bus(In_to_Q, In[6], Q[6])
$6 max_fall_delay_bus(In_to_Q, In[6], Q[6])
```

**Note:**

Avoid using a bus arc in nonbus max/min rise/fall delay functions.

---

## SDF Mapping Limitations

The SDF mapping file syntax has the following limitations:

- You cannot use wildcard characters.
- You cannot specify instance-specific mapping format. An SDF mapping format must be specified for a library cell. The format is applied when writing SDF for every cell that is an instance of that library cell.
- If the SDF mapping file does not provide a format for every cell present in the library, PrimeTime prints these cells using the default format of the `write_sdf` command.

---

## Mapped SDF File Examples

The following examples are different types of mapped SDF files.

### Library File for Cells EXMP and FF1

A complete SDF mapping example for cells EXMP and FF1 is shown in this section. The `min_pulse_width` arcs are labeled by using Style 2.

The following is an example of a library file, `example.lib`.

```
library(example) {define("timing_label_mpw_low", "pin",
"string");
define("timing_label", "timing", "string");
define("timing_label_mpw_low", "pin", "string");

/* If using style #1 to specify min_pulse_width info
*/
/* define("timing_label_mpw_low", "min_pulse_width",
"string");
```

```

*/
/* define("timing_label_mpw_high", "min_pulse_width",
"string");*/

type(two_bit) {
 base_type : array;
 data_type : bit;
 bit_width : 2;
 bit_from : 1;
 bit_to : 0;
 downto : true;
}
cell (EXMP) {
 version : 1.00;
 area : 1.0000000;
 cell_footprint : "EXMP";
 bus(IN) {
 bus_type : two_bit;
 direction : input;
 pin(IN[1:0]) {
 capacitance : 1;
 }
 }

 pin (A) {
 direction : input;
 capacitance : 0.49;
 fanout_load : 0.49;
 max_transition : 4.500000;
 }
 pin (B) {
 direction : input;
 capacitance : 0.51;
 fanout_load : 0.51;
 max_transition : 4.500000;
 }
 pin(Y){
 direction : output;
 capacitance : 0.00;
 function : "(A & B & IN[0] & IN[1])";
 timing () {
 related_pin : "A";
 timing_label : "A_Z";
 rise_transition (transitionDelay){ ...}
 fall_transition (transitionDelay){...}
 rise_propagation (propDelay){...}
 fall_propagation (propDelay){...}
 }
 timing () {
 related_pin : "B";
 timing_label : "B_Z";
 rise_transition (transitionDelay){...}
 fall_transition (transitionDelay){...}
 }
 }
}

```

```

 rise_propagation (propDelay){...}
 fall_propagation (propDelay) { ...}
 }
 timing () {
 related_pin : "IN[0]";
 timing_label : "IN[0]_Z";
 rise_transition (transitionDelay){...}
 fall_transition (transitionDelay){...}{
 rise_propagation (propDelay){...}
 fall_propagation (propDelay){...}
 }
 timing () {
 related_pin : "IN[1]";
 timing_label : "IN[1]_Z";
 rise_transition (transitionDelay){...}
 fall_transition (transitionDelay){...}
 rise_propagation (propDelay){...}
 fall_propagation (propDelay){...}
 }
 ...
}

cell(FD2) {
 area : 9;
 pin(D) {
 direction : input;
 capacitance : 1;
 timing() {
 timing_label : "setup_D";
 timing_type : setup_rising;
 intrinsic_rise : 0.85;
 intrinsic_fall : 0.85;
 related_pin : "CP";
 }
 timing() {
 timing_label : "hold_D";
 timing_type : hold_rising;
 intrinsic_rise : 0.4;
 intrinsic_fall : 0.4;
 related_pin : "CP";
 }
 }
 pin(CP) {
 direction : input;
 capacitance : 1;
 min_period : 5.0
 min_pulse_width_high: 1.5
 min_pulse_width_low: 1.5
 timing_label_mpw_low: "min_pulse_low_CP"
 timing_label_mpw_high: "min_pulse_high_CP"
 }
 pin(CD) {

```

```

 direction : input;
 capacitance : 2;
 timing() {
 timing_label : "recovery_rise_CD";
 timing_type : recovery_rising;
 intrinsic_rise : 0.5;
 related_pin : "CP";
 }
}
....
pin(Q) {
 direction : output;
 function : "IQ";
 internal_node : "Q";
 timing() {
 timing_label : "CP_Q";
 timing_type : rising_edge;
 intrinsic_rise : 1.19;
 intrinsic_fall : 1.37;
 rise_resistance : 0.1458;
 fall_resistance : 0.0523;
 related_pin : "CP";
 }
 timing() {
 timing_label : "clear_Q";
 timing_type : clear;
 timing_sense : positive_unate;
 intrinsic_fall : 0.77; /* CP -> Q intrinsic - 0.6 ns */
 fall_resistance : 0.0523;
 related_pin : "CD";
 }
}
pin(QN) {
 direction : output;
 function : "IQN";
 internal_node : "QN";
 timing() {
 timing_label : "CP_QN";
 timing_type : rising_edge;
 intrinsic_rise : 1.47;
 intrinsic_fall : 1.67;
 rise_resistance : 0.1523;
 fall_resistance : 0.0523;
 related_pin : "CP";
 }
 timing() {
 timing_label : "preset_QN";
 timing_type : preset;
 timing_sense : negative_unate;
 intrinsic_rise : 0.87; /* CP -> QN intrinsic - 0.6 ns */
 rise_resistance : 0.1523;
 related_pin : "CD";
 }
}

```

```

 }
 }
}

```

**Note:**

The `min_pulse_width_low/high` constraints have been defined as attributes inside the pin group. The timing label attributes `timing_label_mpw_low` and `timing_label_mpw_high` are used to refer to these constraints.

**Mapped SDF File**

The following is an example of a mapped SDF file, `example.map`.

```

This is a comment
$SDF_VERSION "OVI 2.1"
$SDF_BUSBIT "[]"
$SDF_CELL EXMP
"(DELAY
 (ABSOLUTE
 (IOPATH $1 $2 ($3::$4) ($5::$6))
 (IOPATH $7 $2 ($8::$9) ($10::$11))
)
)"
$3 min_rise_delay(A_Z)
$4 max_rise_delay(A_Z)
$5 min_fall_delay(A_Z)
$6 max_fall_delay(A_Z)
$1 pin(A)
$2 pin(Z)
$7 bus(IN[0])
$8 min_rise_delay(IN[0]_Z)
$9 max_rise_delay(IN[0]_Z)
$10 min_fall_delay(IN[0]_Z)
$11 max_fall_delay(IN[0]_Z)
$SDF_CELL_END

$SDF_CELL DFF
"(DELAY
 (ABSOLUTE
 (IOPATH (posedge $1) $2 ($3::$4) ($5::$6))
 (IOPATH (posedge $7) $8 ($9::$10) ($11::$12))
 (IOPATH (negedge $13) $2 (::) ($16::$17))
 (IOPATH (negedge $13) $8 ($18::$19) (::))
)
)
(TIMINGCHECK
 (SETUP $20(posedge $1)) ($30::$31))
 (HOLD $20(posedge $1)) ($32::$33))
 (WIDTH (negedge $1) ($34))
 (RECOVERY (posedge $13) (posedge $1)) ($36))
 (PERIOD (posedge $1) ($37))
)"

```

```

$1 pin(CP)
$2 pin(Q)
$3 min_rise_delay(CP_Q)
$4 max_rise_delay(CP_Q)
$5 min_fall_delay(CP_Q)
$6 max_fall_delay(CP_Q)
$7 pin(CP)
$8 pin(QN)
$9 min_rise_delay(CP_QN)
$10 max_rise_delay(CP_QN)
$11 min_fall_delay(CP_QN)
$12 max_fall_delay(CP_QN)
$13 pin(CD)
$16 min_fall_delay(clear_Q)
$17 max_fall_delay(clear_Q)
$18 min_rise_delay(preset_QN)
$19 max_rise_delay(preset_QN)
$20 pin(D)
$30 min_rise_delay(setup_D)
$31 max_rise_delay(setup_D)
$32 min_rise_delay(hold_D)
$33 max_rise_delay(hold_D)
$34 min_fall_delay(min_pulse_low_CP)
$36 max_rise_delay(recovery_rise_CD)
$37 min_period(CP)

```

## Three-State Buffers

This section gives an example of the following files for a three-state noninverting buffer.

- Library file
- Mapped SDF file

### Library File

The following is a detailed example of a library file for a three-state noninverting buffer.

```

/
*-----
Internal Tristate Non-Inverting Buffer, Positive Enable, 1x
Drive
-----*
/
define("timing_label", "timing", "string");
define("timing_label_mpw_low", "pin", "string");
define("timing_label_mpw_low", "pin", "string");

cell(BTS) {
 area : 63 ;
 pin(Z) {
 direction : output ;
 }
}

```

```

function : "A";
max_capacitance : 0.14000 ;
capacitance : 0.00420 ;
three_state : "E'";
timing() {
 related_pin : "A" ;
 timing_sense : positive_unate ;
 timing_label : "A_Z";
 cell_rise(table_1) {
 values (...) ;
 }
 rise_transition(table_1) {
 values (...) ;
 }
 cell_fall(table_1) {
 values (...) ;
 }
 fall_transition(table_1) {
 values (...) ;
 }
 intrinsic_rise : 0.31166 ;
 intrinsic_fall : 0.40353 ;
}
timing() {
 related_pin : "E" ;
 timing_label : "E_Z_3s_enable";
 cell_rise(table_1) {
 values (...) ;
 }
 rise_transition(table_1) {
 values (...) ;
 }
 cell_fall(table_1) {
 values (...) ;
 }
 fall_transition(table_1) {
 values (...) ;
 }
 intrinsic_rise : 0.22159 ;
 intrinsic_fall : 0.27933 ;
}
timing() {
 related_pin : "E" ;
 timing_type : three_state_disable ;
 timing_label : "E_Z_3s_disable";
 cell_rise(table_10) {
 values (...) ;
 }
 rise_transition(table_10) {
 values (...) ;
 }
 cell_fall(table_10) {
 values (...) ;
 }
}

```



```

 }
 fall_transition(table_10) {
 values (...) ;
 }
 intrinsic_rise : 0.30693 ;
 intrinsic_fall : 0.19860 ;
}
}
pin(A) {
 direction : input ;
 capacitance : 0.00423 ;
}
pin(E) {
 direction : input ;
 capacitance : 0.01035 ;
}
}

```

### Mapped SDF File

The following is an example of a mapped SDF file.

```

$SDF_VERSION "OVI 2.1"
$SDF_BUSBIT "[]"
$SDF_CELL BTS
"(DELAY
 (ABSOLUTE
 (IOPATH $1 $3 ($4::$5) ($6::$7))
 (IOPATH $2 $3 ($8::$9) ($10::$11) ($12::$13) ($8::$9)
($14::$15) ($10::$11))
)"
####
$1 pin(A)
$2 pin(E)
$3 pin(Z)
####
$4 min_rise_delay(A_Z)
$5 max_rise_delay(A_Z)
$6 min_fall_delay(A_Z)
$7 max_fall_delay(A_Z)
#####
$8 min_rise_delay(E_Z_3s_enable)
$9 max_rise_delay(E_Z_3s_enable)
$10 min_fall_delay(E_Z_3s_enable)
$11 max_fall_delay(E_Z_3s_enable)
$12 min_rise_delay(E_Z_3s_disable)
$13 max_rise_delay(E_Z_3s_disable)
$14 min_fall_delay(E_Z_3s_disable)
$15 max_fall_delay(E_Z_3s_disable)
####

```



# Index

---

## A

- advanced on-chip variation (see AOCV)
- analysis mode
  - enabling 5-20
  - fast performance 5-20
- annotated checks, removing 8-18
- annotated delay
  - removing 8-17
  - set 8-19
- annotating transition times 8-20
- annotations, setting from the command line 8-18
- AOCV
  - configuring analysis 6-4
  - file format 6-6
  - flow 6-2
  - graph-based solution 6-2
  - importing information 6-4
  - overview 6-2
  - path-based solution 6-3
- arrival window, endpoints 12-53
- Astro
  - change files 3-27
  - ECO format, direct output 3-27
  - hierarchical ECO format 3-27
- asynchronous logic
  - analysis 5-31

- self-timed 5-31
- attributes 4-14, 12-5
  - cell object class, listed 12-6
  - clock object class, listed 12-14
  - defining (define\_user\_attribute) 12-2
  - design object class, listed 12-20
  - importing from .ddc or .db files 12-3
  - lib object class, listed 12-30
  - lib\_cell object class, listed 12-32
  - lib\_pin class, listed 12-35
  - net object class, listed 12-42
  - path\_group object class, listed 12-51
  - pin object class, listed 12-52
  - save 12-5
  - timing path object class, listed 12-95

## B

- back-annotation 4-14, 8-2
  - files 8-2
  - files, improving reading performance 3-5
  - incomplete 9-20
- backtrack limit
  - change 5-27
  - default 5-26
  - prove-false, change 5-27
- bidirectional inout pins 3-3

- block, write physical information for hierarchical 4-11
- borrowing time (latches) 5-2
- buffers, parallel buffer reduction 5-41
- bus contention 5-37
  - steady-state 5-37
  - transient 5-37
- buses
  - three-state 5-37

## C

- calculating
  - cell path depths 6-9
- capacitance
  - in SPEF 9-9
  - lumped 9-10
- carry-bypass adder 5-29
- CCS receiver model
  - for path-based analysis 5-19
  - path-based analysis 5-19
- CCS timing libraries
  - invoking scaling 10-9
  - scaling 10-9
- CCS timing model
  - advantages 10-3
  - overview 10-5
- cell object class attributes list 12-6
- characterization trip points 9-6
- characterize
  - derive annotated delays and parasitics on internal nets 4-6
  - derive clock information 4-4
  - derive constant logic values on inputs 4-5
  - derive design rule checks 4-6
  - derive input and output delay 4-4
  - derive input drive strength and port capacitance 4-5
  - derive point-to-point timing exceptions 4-4
  - derive wire load models 4-5

- characterize\_context command 4-4, 4-5, 4-8, 4-13
- check\_timing command 11-19
- clearing netlist changes 3-26
- clock
  - generated 4-14
  - pins 3-3
  - unrelated 5-35
- clock buffers, parallel 5-41
- clock mesh/spine networks 8-12
- clock object class attributes list 12-14
- clock reconvergence pessimism, time borrowing 5-10
- clock uncertainty
  - defining 5-9
- commands
  - characterize\_context 4-5, 4-8, 4-13
  - check\_timing 11-19
  - complete\_net\_parasitics 9-21
  - connect\_power\_domain 11-27
  - create\_power\_domain 11-26
  - create\_power\_net\_info 11-26
  - create\_supply\_set 11-23
  - current\_design, differences between PrimeTime and Design Compiler 3-3
  - debug\_script 2-14
  - define\_proc\_attributes 2-5
  - define\_scaling\_lib\_group 10-10, 10-13, 11-10
  - define\_user\_attribute 12-2
  - estimate\_eco 3-10
  - get\_\* 11-12
  - get\_attribute 12-2
  - get\_lib\_timing\_arcs 12-106
  - get\_power\_domains 11-12
  - get\_power\_switches 11-13
  - get\_supply\_nets 11-12
  - get\_supply\_ports 11-13
  - get\_supply\_sets 11-13
  - get\_timing\_arcs 12-105
  - get\_timing\_paths 5-14, 5-15, 5-21, 12-103
  - insert\_buffer 3-7, 3-17, 3-22

link\_design 3-8  
list\_attributes 12-2  
load\_upf 11-4  
parse\_proc\_arguments 2-8  
read\_aocvm 6-5  
read\_lib 11-24  
read\_parasitics 3-25, 9-9  
read\_sdc 3-5  
read\_sdf 8-2  
regexp 2-2  
remove\_resistance 9-4  
remove\_annotated\_check 8-17  
remove\_annotated\_delay 8-17  
remove\_annotated\_parasitics 9-23  
remove\_context 4-13  
remove\_coupling\_separation 3-27  
remove\_ideal\_latency 7-4  
remove\_ideal\_network 7-3  
remove\_ideal\_transition 7-4  
remove\_max\_time\_borrow 5-14  
remove\_user\_attribute 12-2  
rename\_cell 3-19  
rename\_net 3-19  
report\_constraint 10-12  
report\_annotated\_delay 8-6  
report\_annotated\_parasitics 9-22  
report\_aocvm 6-5  
report\_attribute 5-17  
report\_attributes 12-2  
report\_delay\_calculation 10-6  
report\_driver\_model 12-107  
report\_lib 5-22  
report\_lib\_groups 10-14  
report\_power\_domain 11-16  
report\_power\_network 11-16  
report\_power\_pin\_info 11-16  
report\_power\_switch 11-17  
report\_supply\_net 11-17  
report\_supply\_set 11-18  
report\_timing 5-14, 5-15, 5-17, 5-21, 6-3, 6-5  
reporting\_power\_supply 11-15  
save\_upf 11-5  
scale\_parasitics 9-11  
set\_annotated\_check 8-19  
set\_annotated\_delay 8-19  
set\_annotated\_transition 8-20  
set\_aocvm\_coefficient 6-10  
set\_clock\_uncertainty 5-9  
set\_coupling\_separation 3-27  
set\_data\_check 5-44  
set\_drive 4-5  
set\_driving\_cell 4-5  
set\_ideal\_latency 7-4  
set\_ideal\_network 7-2  
set\_ideal\_transition 7-4  
set\_isolation 11-7  
set\_level\_shifter 11-8  
set\_level\_shifter\_strategy 11-20  
set\_load 4-5  
set\_max\_time\_borrow 5-13  
set\_operating\_conditions 11-9  
set\_port\_fanout\_number 4-5  
set\_program\_options 5-20  
set\_related\_supply\_net 11-8  
set\_resistance 9-3  
set\_retention 11-7  
set\_retention\_control 11-7  
set\_temperature 11-10  
set\_timing\_derate 6-3, 6-10  
set\_user\_attribute 12-2  
set\_voltage 11-9, 11-30  
set\_wire\_load\_model 4-5  
size\_cell 3-7, 3-16  
source 11-4  
swap\_cell 3-18  
write 8-5  
write\_astro\_changes 3-27  
write\_binary\_aocvm 6-5  
write\_changes 3-7, 3-22, 3-26  
write\_context 4-3, 4-12  
write\_physical\_annotations 4-3, 4-11  
write\_script 8-5, 11-5, 12-5  
write\_sdc 3-5  
write\_sdf 8-10, A-3

- write\_sdf\_constraints 8-20, 8-22
- complete\_net\_parasitics command 9-21
- COND keyword, example of use with IOPATH delays 8-9
- conditional timing arcs 8-8
- configuring AOCV analysis 6-4
- connect\_power\_domain command 11-27
- constrained pin (for data check) 5-44
- constraint file, write SDF format 8-20
- constraint, scaling interpolation 10-12
- context characterization
  - limitations 4-13
  - perform subdesign timing analysis 4-2
  - set synthesis or optimization constraints 4-2
- context information
  - delete 4-13
  - export 4-12
- conventions for documentation 13xv
- converting
  - PG pins 11-3
- create\_power\_domain command 11-26
- create\_power\_net\_info command 11-26
- create\_supply\_set command 11-23
- critical paths
  - viewing 5-16
- CRPR and time borrowing 5-10
- current\_design command 3-3
- customer support 13xvi
- cycle stealing (time borrowing) 5-2

## D

- data checks 5-44
  - clock domains 5-47
  - library-based 5-48
  - nochange check 5-46
- debug\_script command 2-14
- define\_proc\_attributes command 2-5
- define\_scaling\_lib\_group command 10-10, 10-13, 11-10

- define\_user\_attribute command 12-2
- defining
  - clock uncertainty 5-9
- delay
  - extra source 8-3
  - state-dependent example 8-9
- delay back-annotation 8-2
- delay calculation
  - fast multidrive delay analysis 5-40
  - overview 10-2
- delay changes
  - estimating 3-10
- derate tables
  - file format 6-6
  - reading 6-5
  - specifying 6-5
- Design Compiler
  - characterize context as Design Compiler script 4-12
  - differences between pt\_shell and dc\_shell 3-2
- design object class attributes list 12-20
- design rule checks, derive 4-6
- design rule constraints, scaling 10-12
- design, constrain fully 8-22
- driver model parameters 12-107
- driver reduction
  - conditions 5-43
- dynamic loop breaking 5-33

## E

- ECO
  - modifying designs 3-10
  - parasitics 3-25
- ECO flow
  - examples 3-26
- eco\_write\_changes\_prepend\_libname\_to\_libcell variable 3-23
- editing netlists 3-6
- effective capacitance 9-5

Elmore delay 9-4  
 estimate\_eco command 3-10  
 extra source delay 8-3

## F

false paths 5-22  
 fast analysis mode 5-20  
   monitoring 5-21  
 fast multidrive delay analysis 5-40  
 feedback loop breaking 5-33  
 file compression 8-5  
 file format  
   AOCV 6-6  
 file format for AOCV 6-6  
 files  
   back-annotation 8-2  
   back-annotation, improving reading  
     performance 3-5  
   bytecode-compiled 2-13  
   PG Tcl 11-24  
   read RSPF, SPEF 9-9  
   SDF, command to write 8-10  
   Standard Delay Format (SDF) 8-2  
   write SDF format constraint 8-20  
 floating bus 5-37, 5-39  
 flows  
   AOCV 6-2  
   SDF, faster timing updates 8-7

## G

generated clocks 4-14  
 generating  
   PG Tcl file 11-24  
 get\_\* commands 11-12  
 get\_attribute command 12-2, 12-5  
 get\_lib\_timing\_arcs command 12-106  
 get\_power\_domains command 11-12  
 get\_power\_switches command 11-13

get\_supply\_nets command 11-12  
 get\_supply\_ports command 11-13  
 get\_supply\_sets command 11-13  
 get\_timing\_arcs command 12-105  
 get\_timing\_paths command 5-14, 5-15, 5-21,  
   12-103  
 graph-based AOCV 6-5  
 graph-based solution  
   AOCV 6-2  
 grouping paths for synthesis 3-6

## H

hierarchical ECO format 3-27  
 high capacity mode  
   monitoring 5-20  
 high performance  
   analysis mode 5-20

## I

IC Compiler  
   ECO flow examples 3-26  
 ideal latency  
   removing 7-4  
   using 7-4  
 ideal network  
   introduction 7-2  
   propagating properties 7-2  
   removing 7-3  
   using 7-2  
 ideal transition  
   removing 7-4  
   using 7-4  
 importing  
   AOCV informaiton 6-4  
 incremental timing 9-19  
 inherited loop breaking 5-34  
 inout pins (bidirectional) 3-3  
 input delay and port capacitance, calculate 4-8

insert\_buffer command 3-7, 3-17, 3-22

introduction to AOCV 6-2

IOPATH keyword, example 8-8

## L

latch, time borrowing 5-2

latch-based designs 5-2

level shifters

    multirail cell support 10-12

lib object class attributes list 12-30

lib\_cell object class attributes list 12-32

lib\_pin class attributes list 12-35

lib\_thresholds\_per\_lib variable 9-7

library

    data 12-107

Library Compiler

    multiple voltages 11-21

library\_pg\_file\_pattern variable 11-24

limitations, context characterization 4-13

link\_design command 3-8

link\_path\_per\_instance variable 11-10

list\_attributes command 12-2

load\_upf command 11-4

loop breaking 5-33

    dynamic 5-33

    inherited from .ddc or .db 5-34

lp\_default\_ground\_pin\_name variable 11-25

lp\_default\_power\_pin\_name variable 11-25

lumped 9-10

## M

messages

    default limit 9-11

Miller Effect 10-4

modes 4-14

modifying

    ECO designs 3-10

monitoring

fast analysis mode 5-21

high capacity mode 5-20

multidrive analysis mode 5-41

multidrive delay analysis 5-40

multirail cells

    support 10-12

multivoltage analysis 11-2

## N

net object class attributes list 12-42

net resistance

    override internally estimated value 9-3

    set 9-3

netlist

    editing 3-6

    saving changes 3-7

nets

    internal, derive annotated delays and  
    parasitics 4-6

NLDM

    overview 10-3

nonsequential setup/hold checking 5-44

normalizing multidriven arcs for simulation  
    8-14

## O

output format 3-22

overview

    advanced topics 1-1

    AOCV 6-2

    manual 1-2

## P

parallel arc paths 5-21

parallel clock buffers 5-41

parallel driver reduction 5-43

parasitic values, scaling 9-11

parasitics



- annotate detailed 9-5
- annotate lumped 9-3
- annotate reduced 9-4
- data files 9-22
- environment variables for reduced and detailed 9-6
- export annotated 4-11
- incomplete 9-20
- read file and annotate 9-9
- remove annotated 9-23
- report (verify) annotated 9-22
- scaling values 9-11
- parse\_proc\_arguments command 2-8
- path
  - detected, justify 5-24
  - true, detect most critical 5-24, 5-26
- path groups 3-6
  - worst violation in 3-6
- path type
  - recalculated 5-17
- path\_group object class attributes list 12-51
- path-based analysis 5-15
- path-based analysis, integrated with
  - report\_timing command 5-18
- path-based AOCV solution 6-3
- path-based timing analysis 5-14
- pba\_exhaustive\_endpoint\_path\_limit variable 5-18
- pba\_recalculate\_full\_path variable 5-17
- performing submodule timing 4-2
- PG pins
  - automatically converting 11-3
  - specifying information 11-3
- PG Tcl file 11-24
- pi model (RC) 9-4
- pin
  - clock 3-3
  - object class attributes list 12-52
- point-to-point exceptions 4-13
- PORT construct 8-13
- power and ground pins

- specifying names 11-25
- power domains 11-26
- power supply network
  - reporting commands 11-15
- power\_domains\_compatibility variable 11-26
- procedure
  - changing aspects 2-5
  - defining attributes 2-5
  - modifying existing 2-5

## R

- random coefficients 6-9
- RC 9-3, 9-4
  - detailed 9-5
  - meshed 9-5
  - pi model 9-4
  - reduced 9-4
  - threshold variables 9-8
- rc\_driver\_count\_threshold\_for\_fast\_multidrive\_analysis variable 5-40
- rc\_input\_threshold\_pct\_fall variable 9-8
- rc\_input\_threshold\_pct\_rise variable 9-8
- rc\_output\_threshold\_pct\_fall variable 9-8
- rc\_output\_threshold\_pct\_rise variable 9-8
- rc\_slew\_\* variables 9-8
- rc\_slew\_derate\_from\_library variable 9-8
- rc\_slew\_lower\_threshold\_pct\_fall variable 9-8
- rc\_slew\_lower\_threshold\_pct\_rise variable 9-8
- rc\_slew\_upper\_threshold\_pct\_fall variable 9-8
- rc\_slew\_upper\_threshold\_pct\_rise variable 9-8
- read\_aocvm command 6-5
- read\_lib command 11-24
- read\_parasitics command 3-25, 9-9
- read\_sdf command 8-2
- reading
  - back-annotated delays 8-2
  - SDF file 8-2
- recalculating

- path-based 5-16
- Reduced Standard Parasitic Format
  - reading files 9-9
- reducing parallel clock buffers 5-41
- reducing pessimism
  - for critical paths 5-15
- reducing SDF for clock mesh/spine networks 8-12
- regexp command 2-2
- regular expressions
  - anchor 2-4
  - buses 2-4
- regular expressions 2-2
- related pin (for data check) 5-44
- remove\_annotated\_check command 8-17
- remove\_annotated\_delay command 8-17
- remove\_annotated\_parasitics command 9-23
- remove\_context command 4-13
- remove\_coupling\_separation command 3-27
- remove\_ideal\_latency command 7-4
- remove\_ideal\_network command 7-3
- remove\_ideal\_transition command 7-4
- remove\_max\_time\_borrow command 5-14
- remove\_resistance command 9-4
- remove\_user\_attribute command 12-2
- removing annotated checks 8-18
- removing annotated delays 8-17
- removing annotated delays and checks 8-17
- rename\_cell command 3-19
- rename\_net command 3-19
- report\_annotated\_delay command 8-6
- report\_annotated\_parasitics command 9-22
- report\_aocvm command 6-5
- report\_attribute command 5-17, 12-2
- report\_constraint command 10-12
- report\_delay\_calculation command 10-6
- report\_driver\_model command 12-107
- report\_lib command 5-22
- report\_lib\_groups command 10-14

- report\_power\_domain command 11-16
- report\_power\_network command 11-16
- report\_power\_pin\_info command 11-16
- report\_power\_switch command 11-17
- report\_supply\_net command 11-17
- report\_supply\_set command 11-18
- report\_timing command 5-14, 5-15, 5-17, 5-21, 6-3, 6-5
- reporting commands
  - power supply network 11-15
- reporting delay back-annotation 8-6
- reporting net delays 8-6
- reports, generate custom 12-103
- resistance and capacitance (RC) 9-4
- retain arcs 5-22
  - clock-to-output 5-22
- RSPF
  - reading files 9-9
- runtime
  - cause of long 8-22
  - improve 5-35

## S

- save\_upf command 11-5
- saving changes
  - netlist 3-7
- scale\_parasitics command 9-11
- scaling
  - design rule constraints 10-12
  - guidelines 10-10
  - invoking for voltage and temperature 10-9
- scaling interpolation for constraints 10-12
- scaling parasitic values 9-11
- scaling support
  - multirail cells 10-12
- SDC (Synopsys Design Constraints) 3-5
- SDF 4-11, 8-6, 9-3
  - (Standard Delay Format) 3-6
  - output versions 8-10

- reducing for clock mesh/spine networks 8-12
- supported constructs 8-10
- SDF mapping
  - timing arcs A-2
  - using A-3
- sdf\_align\_multi\_drive\_cell\_arcs variable 8-12
- sdf\_align\_multi\_drive\_cell\_arcs\_threshold variable 8-12
- sdf\_enable\_port\_construct variable 8-12
- sdf\_enable\_port\_construct\_threshold variable 8-12
- searching
  - parallel arc paths 5-21
- self-timed asynchronous logic 5-31
- set\_annotated\_check command 8-19
- set\_annotated\_delay command 8-19
- set\_annotated\_transition command 8-20
- set\_aocvm\_coefficient command 6-10
- set\_clock\_transition command 3-3
- set\_clock\_uncertainty command 5-9
- set\_coupling\_separation command 3-27
- set\_data\_check command 5-44
  - commands
    - set\_data\_check 5-47
- set\_drive command 4-5
- set\_driving\_cell command 4-5
- set\_ideal\_latency command 7-4
- set\_ideal\_network command 7-2
- set\_ideal\_transition command 7-4
- set\_isolation command 11-7
- set\_level\_shifter command 11-8
- set\_level\_shifter\_strategy command 11-20
- set\_load command 4-5
- set\_max\_time\_borrow command 5-13
- set\_operating\_conditions command 11-9
- set\_port\_fanout\_number command 4-5
- set\_program\_options
  - enable\_high\_capacity 5-20
- set\_program\_options command 5-20
- set\_related\_supply\_net command 11-8
- set\_resistance command 9-3
- set\_retention command 11-7
- set\_retention\_control command 11-7
- set\_temperature command 11-10
- set\_timing\_derate command 6-3, 6-10
- set\_user\_attribute command 12-2
- set\_voltage command 11-9, 11-30
- set\_wire\_load\_model command 4-5
- setting
  - annotations from the command line 8-18
  - optimization constraints 4-2
  - synthesis constraints 4-2
- setup/hold pessimism reduction
  - hold-preferred improvement mode 5-49
  - optimization constraint 5-50
  - optimization mechanism 5-50
  - setup-preferred improvement mode 5-49
  - total negative slack improvement mode 5-50
  - use model 5-49
  - user interface 5-51
- sh\_limited\_messages variable 9-11
- sh\_message\_limit variable 9-11
- sharing scripts 3-4
- short path borrowing 3-2
- SHPR
  - (see setup/hold pessimism reduction)
- size\_cell command 3-7, 3-16
- slack 4-14, 4-16, 4-17, 5-28
- SNPS\_TCLPRO\_HOME variable 2-14
- SolvNet
  - accessing 13xvi
  - documentation 13xiv
  - Download Center 13xiv
- source command 11-4
- Standard Delay Format (SDF) 3-6, 8-2
- Standard Parasitic Exchange Format (SPEF) 9-6
- state dependent delays example 8-9
- steady-state bus contention 5-37

- stealing cycles (time borrowing) 5-2
- subdesign
  - derive the context 4-3
  - export timing and physical information 4-3
- supply nets 11-23
- supply sets 11-23
- swap\_cell command 3-18
- Synopsys Design Constraints (SDC) 3-5

## T

### Tcl

- advanced features 2-1
- autoload 2-2
- packages 2-2
- parsing arguments 2-8
- TclPro 2-9

### TclPro

- bytecode-compiled files 2-13
- limitations 2-14
- procheck 2-9
- procomp 2-9, 2-13
- prodebug 2-14
- prodebug 2-9

- temperature, invoking scaling 10-9
- three\_state\_disable arc 5-38
- three\_state\_disable timing arcs 5-37
- three\_state\_enable timing arcs 5-37
- three-state bus 5-37
- time borrowed from endpoint or startpoint 5-11
- time borrowing (latches) 5-2
  - limiting 5-13
  - maximum limit 5-14
  - negative 5-12
  - short path 3-2
  - time borrowed from endpoint 5-11
  - time given to startpoint 5-11
- time given to startpoint or endpoint 5-11
- timing arcs
  - reducing number 5-41

- three\_state\_disable 5-37
- three\_state\_enable 5-37
- timing checks, setting 8-19
- timing exceptions 4-13
- timing path object class attributes list 12-95
- timing\_aocvm\_analysis\_mode variable 6-4, 6-9
- timing\_aocvm\_enable\_analysis variable 6-2, 6-4
- timing\_disable\_bus\_contention\_check variable 5-36, 5-39
- timing\_disable\_floating\_bus\_check variable 5-36, 5-40
- timing\_dynamic\_loop\_breaking variable 5-33
- timing\_early\_launch\_at\_borrowing\_latches disabled 5-11
- variable 5-11
- timing\_keep\_loop\_breaking\_disabled\_arcs variable 5-34
- timing\_prelayout\_scaling variable 11-11
- timing\_propagate\_through\_unlocked\_registers variable 5-48
- timing\_reduce\_multi\_drive\_net\_arcs\_threshold variable 5-43
- timing\_reduce\_multi\_driven\_net\_arcs variable 5-43
- timing\_remove\_clock\_reconvergence\_pessimism variable 5-9
- timing\_report\_recalculation\_status variable 5-18
- timing\_report\_use\_worst\_parallel\_cell\_arc variable 5-21
- timing\_save\_pin\_arrival\_and\_slack variable 12-53
- timing\_slew\_propagation\_mode variable 5-14
- tracing
  - parallel arc paths 5-21
- transient bus contention 5-37
- transparent latch, time borrowing 5-2
- trip points 9-6
- true path reporting 5-24

- accelerate 5-28
- define a threshold 5-28
- true\_delay\_prove\_false\_backtrack\_limit
  - variable 5-27
- true\_delay\_prove\_true\_backtrack\_limit
  - variable 5-27

## U

- unateness
  - see the Design Compiler Reference Manual*
- undefined generated clocks
  - supporting multiple levels 5-48
- uniquification of modified blocks 3-8
- UNIX gzip 8-5
- unrelated clocks 5-35
- updating
  - PG pins 11-3
- UPF commands
  - using in PrimeTime 11-7

## V

- variable
  - timing\_disable\_floating\_bus\_check 5-40
- variables
  - eco\_write\_changes\_prepend\_libname\_to\_libcell 3-23
  - for reduced and detailed parasitics 9-6
  - lib\_thresholds\_per\_lib 9-7
  - library\_pg\_file\_pattern 11-24
  - link\_path\_per\_instance 11-10
  - lp\_default\_ground\_pin\_name 11-25
  - lp\_default\_power\_pin\_name 11-25
  - pba\_exhaustive\_endpoint\_path\_limit 5-18
  - pba\_recalculate\_full\_path 5-17
  - power\_domains\_compatibility 11-26
  - RC threshold 9-8
  - rc\_driver\_count\_threshold\_for\_fast\_multidrive\_analysis 5-40
  - rc\_input\_threshold\_pct\_rise 9-8
  - rc\_output\_threshold\_pct\_fall 9-8

- rc\_output\_threshold\_pct\_rise 9-8
- rc\_slew\_ 9-8
- rc\_slew\_derate\_from\_library 9-8
- rc\_slew\_lower\_threshold\_pct\_fall 9-8
- rc\_slew\_lower\_threshold\_pct\_rise 9-8
- rc\_slew\_upper\_threshold\_pct\_rise 9-8
- sdf\_align\_multi\_drive\_cell\_arcs 8-12
- sdf\_align\_multi\_drive\_cell\_arcs\_threshold 8-12
- sdf\_enable\_port\_construct 8-12
- sdf\_enable\_port\_construct\_threshold 8-12
- sh\_limited\_messages 9-11
- sh\_message\_limit 9-11
- SNPS\_TCLPRO\_HOME 2-14
- timing\_allow\_short\_path\_borrowing 3-3
- timing\_aocvm\_analysis\_mode 6-4, 6-9
- timing\_aocvm\_enable\_analysis 6-2, 6-4
- timing\_disable\_bus\_contention\_check 5-36, 5-39
- timing\_disable\_floating\_bus\_check 5-36
- timing\_dynamic\_loop\_breaking 5-33
- timing\_early\_launch\_at\_borrowing\_latches 5-11
- timing\_keep\_loop\_breaking\_disabled\_arcs 5-34
- timing\_prelayout\_scaling 11-11
- timing\_propagate\_through\_unclocked\_registers 5-48
- timing\_reduce\_multi\_drive\_net\_arcs\_threshold 5-43
- timing\_reduce\_multi\_driven\_net\_arcs 5-43
- timing\_remove\_clock\_reconvergece\_pessimism 5-9
- timing\_report\_recalculation\_status 5-18
- timing\_report\_use\_worst\_parallel\_cell\_arc 5-21
- timing\_save\_pin\_arrival\_and\_slack 12-53
- timing\_slew\_propagation\_mode 5-14
- true\_delay\_backtrack\_limit 5-27
- true\_delay\_prove\_false\_backtrack\_limit 5-27
- viewing
  - critical paths 5-16
- virtual power network 11-7

voltage  
  invoking scaling 10-9

## W

wire load model, derive 4-5  
write command 8-5  
write\_astro\_changes command 3-27  
write\_binary\_aocvm command 6-5

write\_changes command 3-7, 3-22, 3-26  
write\_context command 4-3, 4-12  
write\_physical\_annotations command 4-3,  
  4-11  
write\_script command 8-5, 11-5, 12-5  
write\_sdf command 8-10, A-3  
write\_sdf\_constraints command 8-20, 8-22  
writing Astro change files 3-27