ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Top-down digital design flow

EDA tools:
Mentor Modelsim
Synopsys Design Compiler
Cadence SoC Encounter

Alain Vachoux
Microelectronic Systems Lab
STI-IMM-LSM
alain.vachoux@epfl.ch

Version 3.1 (November 2006)

Document history

Version	Date	Notes
1.0	2003	Initial version.
2.0.1	23 apr 04	Updated for new EDA tool releases. First design example (adder-subtractor) only.
2.0.2	26 may 04	Minor changes.
2.0.3	3 jun 04	Minor changes.
3.0.1	11 oct 05	Use Cadence Encounter as P&R tool.
3.0.2	6 dec 05	Minor issues fixed.
3.1	Nov 2006	PaR chapter with SoC Encounter rewritten.

Table of Contents

Chapter 1: Introduction

This document details the typical steps of a top-down digital VHDL/Verilog design flow with the help of one simple design example.

The following tools are considered in this document:

- Modelsim v6.1b or higher, from Mentor Graphics.
- Design Compiler and Design Vision 2005.09 or higher from Synopsys.
- Encounter 4.1 and IC 5.0.33 or higher from Cadence Design Systems.

The design kit used is the Hit-Kit 3.70 from AMS. The process is the 0.35 micron 4-metal CMOS called C35B4.

Each of the next chapters in this document is addressing a specific set of tasks. Chapter 2 is about VHDL and Verilog simulation, chapter 3 is about logic synthesis and chapter 4 is about place and route. Steps in these chapters are not necessarily to be done in the given sequence. Go to “1.7 Design flow steps” to get a typical sequence of steps supporting a top-down approach.

1.1 Top-down design flow

Figure 1.1: Top-down design flow.

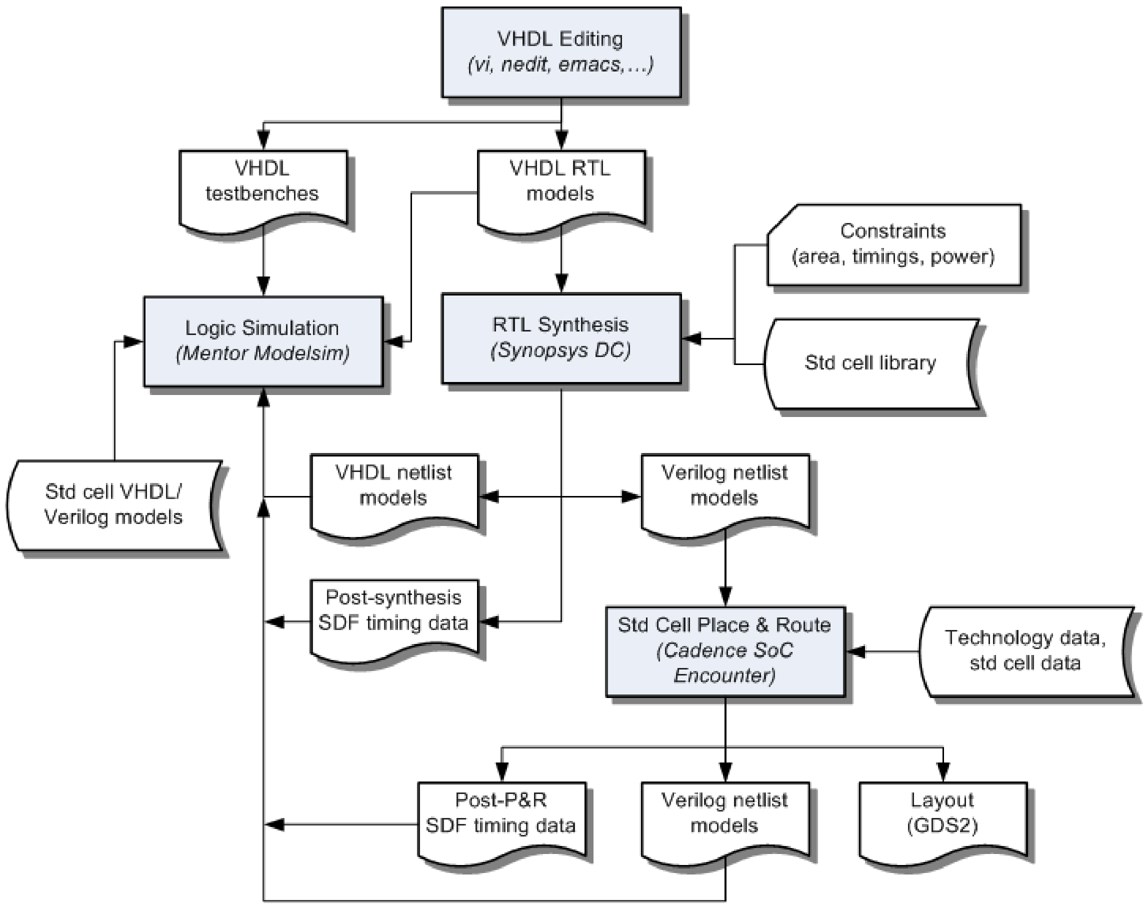

VHDL Editing
(vi, nedit, emacs,...)
VHDL testbenches
VHDL RTL models
Constraints (area, timings, power)
Logic Simulation (Mentor Modelsim)
RTL Synthesis (Synopsys DC)
Std cell library
Std cell VHDL/ Verilog models
VHDL netlist models
Verilog netlist models
Post-synthesis SDF timing data
Std Cell Place & Route (Cadence SoC Encounter)
Technology data, std cell data
Post-P&R SDF timing data
Verilog netlist models
Layout (GDS2)

Figure 1.1 illustrates the top-down flow that includes the following steps:

- **VHDL RTL model creation**

  The goal here is to develop synthesizable VHDL models at the RTL level (RTL means Register-Transfer Level). Such models usually define a clear separation between control parts (e.g. finite state machines - FSM) and operative parts (e.g. arithmetic and logic units). Registers are used to store small size data between clock cycles. RAM/ROM memories are used to store large amounts of data or program code. Blocks such as FSMs, ALUs, registers are usually described as behavioural models that do not imply any particular gate-level implementation. Tools used at this step can range from simple text editors to dedicated graphical environments that generate VHDL code automatically.

- **RTL simulation**

  The VHDL RTL models are validated through simulation by means of a number of testbenches also written in VHDL.

- **RTL synthesis**

  The synthesis process infers a possible gate-level realisation of the input RTL description that meets user-defined constraints such as area, timings or power consumption. The design constraints are defined outside the VHDL models by means of tool-specific commands. The targetted logic gates belong to a library that is provided by a foundry or an IP company as part of a so-called design kit. Typical gate libraries include a few hundreds of combinational and sequential logic gates. Each logic function is implemented in several gates to accomodate several fanout capabilities or drive strengths. The gate library is described in a tool-specific format that defines, for each gate, its function, its area, its timing and power characteristics and its environmental constraints.
  The synthesis step generates several outputs: a gate-level VHDL netlist, a Verilog gate-level netlist, and a SDF description. The first netlist is typically used for post-synthesis simulation, while the second netlist is better suited as input to the place&route step. The SDF description includes delay information for simulation. Note that considered delays are at this step correct for the gates but only estimated for the interconnections.

- **Post-synthesis gate-level simulation**

  The testbenches used for RTL model validation can be reused (with possibly some modifications to use the VHL gate-level netlists). The gate-level simulation makes use of VHDL models for the logic gates that are provided in the design kit. These VHDL models follow the VITAL modelling standard to ensure proper back-annotation of delays through the SDF files generated by the synthesis or the place&route step.

- **Standard cell place and route**

  The place&route (P&R) step infers a geometric realisation of the gate-level netlist so-called a layout. The standard cell design style puts logic cells in rows of equal heights. As a consequence, all logic gates in the library have the same height, but may have different widths. Each cell has a power rail at its top and a ground rail at its bottom.
  The interconnections between gates are today usually done over the cells since current processes allow several metal layers (i.e. 4 metal layers for the AMS C35 process). As a consequence, the rows may be abutted and flipped so power and ground rails are shared between successive rows.
  The P&R step generates several outputs: a geometric description (layout) in GDS2 format, a SDF description and a Verilog gate-level netlist. The SDF description now includes interconnect delay. The Verilog netlist may be different from the one read as input as the P&R step may make further timing optimisations during placement, clock tree generation and routing (e.g. buffer insertion).
  Post-layout gate-level simulation

  The Verilog gate-level netlist can be simulated by using the existing VHDL testbenches and the more accurate SDF data extracted from the layout.

- **System-level integration**

  The layout description is then integrated as a block in the designed system. This step is not covered in this document.

1.2 Design project organisation

Given the number of EDA tools and files used in the flow, it is strongly recommended to organise the working environment in a proper way. To that end, the create_eda_project script can be used to create a directory structure in which design files will be stored. The use of the script is as follows:

```
create_eda_project <project-name>
```

where <project-name> is the name of the top-level directory that will host all design files for the projects.

For example, to create the project directory called ADDSUB that will be used to do the tasks presented in the rest of this document, execute the following command:

```
[27]vachoux@lsmsun1-educ> create_eda_project ADDSUB
```

The ADDSUB top-level directory hosts the configuration files for logic simulation (Modelsim), logic synthesis (Synopsys DC) and standard cell place and route (Cadence SoC Encounter). As a consequence, ***it is required that the tools are always started from that point***. One exception is full-custom layout tools (Cadence IC) that must be started from the subdirectory LAY, which hosts different configuration files.

Figure 1.2 gives the proposed directory structure and the role of each subdirectory (directories have a "/" at the end of their names). Text starting with "#" is a comment. The actual use of the subdirectories and files will be explained while going throughout the tutorial in this document.

Figure 1.2: Design project structure.

```
<project-name>/          # project directory home
   .synopsys_dc.setup    # setup file for Synopsys tools
   modelsim.ini          # setup file for Modelsim tool
   DOC/                  # documentation (pdf, text, etc.)
   HDL/                  # VHDL/Verilog source files
      GATE/              # gate-level netlists
      RTL/               # RTL descriptions
      TBENCH/            # testbenches
   IP/                   # external blocks (e.g., memories)
   LAY/                  # full-custom layout files
   LIB/                  # design libraries
      MSIM/              # Modelsim library (VHDL, Verilog)
      SNPS/              # Synopsys library (VHDL, Verilog)
   PAR/                  # place & route files
      BIN/               # commands, scripts
      CONF/              # configuration files
      CTS/               # clock tree synthesis files
      DB/                # database files
      DEX/               # design exchange files
      LOG/               # log files
      RPT/               # report files
      SDC/               # system design constraint files
      TEC/               # technology files
      TIM/               # timing files
   SIM/                  # simulation files
      BIN/               # commands, scripts
      OUT/               # output files (e.g., waveforms)
   SYN/                  # synthesis files
      BIN/               # commands, scripts
      DB/                # database files
      RPT/               # report files
      SDC/               # system design constraint files
      TIM/               # timing files
   TST/                  # test files
      BIN/               # commands, scripts
      RPT/               # report files
      TV/                # test vectors
```

1.3 EDA tools and design kit configuration

In order to use the EDA tools and the design kit, a file called edadk.conf must exist either in your home directory or in the directory from which the tools are launched (the top-level project directory. This file may exist in both directories, the content of the edadk.conf file in the working directory superseding the definitions in the same file in the home directory.

The edadk.conf file must include the following lines (order is not important):

```
mgc msim 6.2c
snps syn 2005.09
cds soce 4.1
cds ic 5.1.41
dk ams hk370
```

This file is recognized on the immsunsrv1 and immsunsrv2 Linux machines only.

1.4 Installation of the AMS design kit

To install the files required by the AMS design kit, execute the ams_setup script in the top-level project directory as follows:

```
ams_setup -p <process> -t <toolset>
```

where:

```
-p    select process
-t    select toolset
```

Running the command without arguments displays a short help.

To install the 0.35 micron CMOS called C35B4 for logic simulation and synthesis, use:

```
[28]vachoux@lsmsun1-ADDSUB> ams_setup -p c35b4 -t synopsys_dc
```

The command produces the following output[1]:

```
IMPORTANT NOTICE

   All AMS documentation and design files made available with this command
   are subject to non-disclosure agreements between Europractice, AMS
   and EPFL, and hence must be considered as strictly confidential.
   For more information: Alain Vachoux, STI-LSM, alain.vachoux@epfl.ch.

-- Configuration parameters:
   AMS HIT-Kit version v3.70
   Process: c35b4
   Tool   : synopsys_dc
-- Configuration log:
File "./.synopsys_dc.setup" created
File "./modelsim.ini" created
Link "LIB/C35_CORELIB_VLOG" to ".../c35_corelib_vlog" created
Link "LIB/C35_IOLIB_3M_VLOG" to ".../c35_iolib_3m_vlog" created
Link "LIB/C35_IOLIB_4M_VLOG" to ".../c35_iolib_4m_vlog" created
Directory "./LIB/SNPS" created
Directory "./LIB/MSIM" created
Modelsim library WORK mapped to ./LIB/MSIM
Synopsys library WORK mapped to ./LIB/SNPS
```

The command creates configuration files for Modelsim and Synopsys as well as the required design libraries for hosting compiled VHDL/Verilog models and for simulation.

1. The output is slightly different if the project directory structure has not been created first.

To install the same design kit for standard cell place and route, use:

```
[29]vachoux@lsmsun1-ADDSUB> ams_setup -p c35b4 -t cadence_soce
```

The command produces the following output:

```
   IMPORTANT NOTICE

   All AMS documentation and design files made available with this command
   are subject to non-disclosure agreements between Europractice, AMS
   and EPFL, and hence must be considered as strictly confidential.
   For more information: Alain Vachoux, STI-LSM, alain.vachoux@epfl.ch.

-- Configuration parameters:
   AMS HIT-Kit version v3.70
   Process: c35b4
   Tool   : cadence_soce
-- Configuration log:
Run ams_encounter in a temporary directory...
Move files from temporary directory to proper location...
../PAR/BIN/gemma.tcl
../PAR/BIN/fillcore.tcl
../PAR/BIN/fillperi.tcl
../PAR/CONF/c35b4_std.conf
../PAR/CTS/ctgen.const
../PAR/DEX/gds2.map
../PAR/SDC/corners.io
```

The command creates files in specific locations in the PAR (place and route) subdirectory.

For information on the AMS design kits, execute the amshk_doc command. It will start a web browser and display the home page of the local AMS web site. Note that some information such as design rules document is password protected.

1.5 VHDL example: Adder-subtractor

The adder-subtractor example will be used as the reference design throughout the topdown flow.

Code 1.1 gives the VHDL model of a generic N bit adder-subtractor. The model is deliberately quite abstract to show how the synthesiser will be able to infer different adder architectures (ripple carry or carry look-ahead architectures) depending on the given constraints.

Code 1.1. RTL synthesisable model of a N bit adder-subtractor.

```
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity addsub is
     generic (NBITS: natural := 4);
     port (
        clk, rst, add: in  std_logic;
        a, b: in  unsigned(NBITS-1 downto 0);
        z: out unsigned(NBITS-1 downto 0));
end entity addsub;

architecture dfl of addsub is
     signal a_reg, b_reg, z_reg: unsigned(NBITS-1 downto 0);
begin
     process (rst, clk)
     begin
        if rst = '1' then
           a_reg <= (others => '0');
           b_reg <= (others => '0');
           z <= (others => '0');
        elsif clk'event and clk = '1' then
           a_reg <= a;
           b_reg <= b;
           z <= z_reg;
        end if;
     end process;
     z_reg <= a_reg + b_reg when add = '1' else
              a_reg - b_reg;
end dfl;
```

To install the VHDL model and its associated testbenches in the project directory, enter the following command in the top-level project directory ADDSUB:

```
[30]vachoux@lsmsun1-ADDSUB> install_topdown addsub
```

The following files are copied:

```
File HDL/RTL/addsub_dfl.vhd created
File HDL/TBENCH/tb_addsub_mapped.vhd created
File HDL/TBENCH/tb_addsub_par.vhd created
File HDL/TBENCH/tb_addsub_rtl.vhd created
File SYN/BIN/addsub_syn.tcl created
File PAR/BIN/addsub_par.tcl created
File PAR/BIN/addsub_nbits8.io created
```

Code 1.2 gives the testbench for the RTL model with N=8 (file tb_addsub_rtl.vhd). It is by far not complete as it only checks one addition and one subtraction. The testbenches for the mapped design (file tb_addsub_mapped.vhd) and for the simulation of the placed and routed Verilog netlist (file tb_addsub_par.vhd) only differs by the instantiation of the component under test. The addsub_nbits8.io file defines the positions of the IO pins for place and route.

Code 1.2. VHDL testbench for the RTL model with N=8.

```
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity tb_addsub_rtl is end;

architecture bench of tb_addsub_rtl is

     constant CLK_PER: time    := 20 ns;
     constant NBITS  : natural := 8;

     signal clk     : std_logic := '0';
     signal rst, add: std_logic;
     signal a, b, z : unsigned(NBITS-1 downto 0)

begin

     UUT: entity work.addsub(dfl)
            generic map (NBITS)
            port map (clk, rst, add, a, b, z);

     clk <= not clk after CLK_PER/2;
     rst <= '0', '1' after CLK_PER/4, '0' after 3*CLK_PER/4;
     add <= '1', '0' after 11*CLK_PER/4;

     stimulus: process
     begin
        wait for 3*CLK_PER/4;
        a <= to_unsigned(31, a'length);
        b <= to_unsigned(12, b'length);
        wait;
     end process stimulus;

end architecture bench;
```

The file addsub_syn.tcl in directory SYN/BIN is a Tcl script that performs synthesis of the VHDL model in batch mode. The file addsub_par.tcl in directory PAR/BIN is a Tcl script that performs the placement and routing of the synthesized Verilog netlist in batch mode.

1.6 Text editing

The edt alias calls the nedit text editor which provides syntax highlighting. It is also possible to use any other convenient editor such as vi, vim, emacs, xemacs, etc. The Modelsim graphical environment has its own text editor with syntax highlighting.

1.7 Design flow steps

Here are the main steps of the top-down design flow with references to the sections in the document that give more details.

Step 1) VHDL model editing (tool: text editor)

Step 2) Pre-synthesis VHDL simulation (tool: Modelsim) *[2.2]*
- 2.1) Compilation of the RTL VHDL model and related testbench
- 2.2) Simulation of the RTL VHDL model

Step 3) Logic synthesis (tool: Synopsys Design Compiler)
- 3.1) RTL VHDL model analysis *[3.2]*
- 3.2) Design elaboration (generic synthesis) *[3.3]*
- 3.3) Design environment definition (operating conditions, wire load model) *[3.4]*
- 3.4) Design constraint definitions (area, clock, timings) *[3.5]*
- 3.5) Design mapping and optimization (mapping to gates) *[3.6]*
- 3.6) Report generation *[3.7]*
- 3.7) VHDL gate-level netlist generation *[3.8]*
- 3.8) Post-synthesis timing data (SDF) generation for the VHDL netlist *[3.8]*
- 3.9) Verilog gate-level netlist generation *[3.8]*
- 3.10) Design constraints generation for placement and routing *[3.9]*

Step 4) Post-synthesis VHDL simulation (tool: Modelsim) *[2.3]*
- 4.1) Compilation of the VHDL/Verilog netlist and related testbench
- 4.2) Simulation of the post-synthesis gate-level netlist with timing data

Step 5) Placement and routing (tool: Cadence Encounter)
- 5.1) Design import (technological data + Verilog netlist) *[4.2]*
- 5.2) Floorplan specification *[4.3]*
- 5.3) Power ring/stripe creation and routing *[4.4]*
- 5.4) Global net connections definition *[4.5]*
- 5.5) CAP cell placement *[4.6]*
- 5.6) Operating conditions definition *[4.7]*
- 5.7) Core cell placement *[4.8]*
- 5.8) Post-placement timing analysis *[4.9]*
- 5.9) Clock tree synthesis (optional) *[4.10]*
- 5.10) Design routing *[4.11]*
- 5.11) Post-route timing optimization and analysis *[4.12]*
- 5.12) Filler cell placement *[4.13]*
- 5.13) Design checks *[4.14]*
- 5.14) Report generation *[4.15]*
- 5.15) Post-route timing data extraction *[4.16]*
- 5.16) Post-route netlist generation *[4.17]*
- 5.17) GDS2 file generation *[4.18]*
- 5.18) Design import in Virtuoso *[4.19]*

Step 6) Post-layout VHDL/Verilog simulation (tool: Modelsim) *[2.4]*
- 6.1) Compilation of the Verilog netlist and related testbench
- 6.2) Simulation of the post-synthesis or post-PaR gate-level netlist with PaR timing data

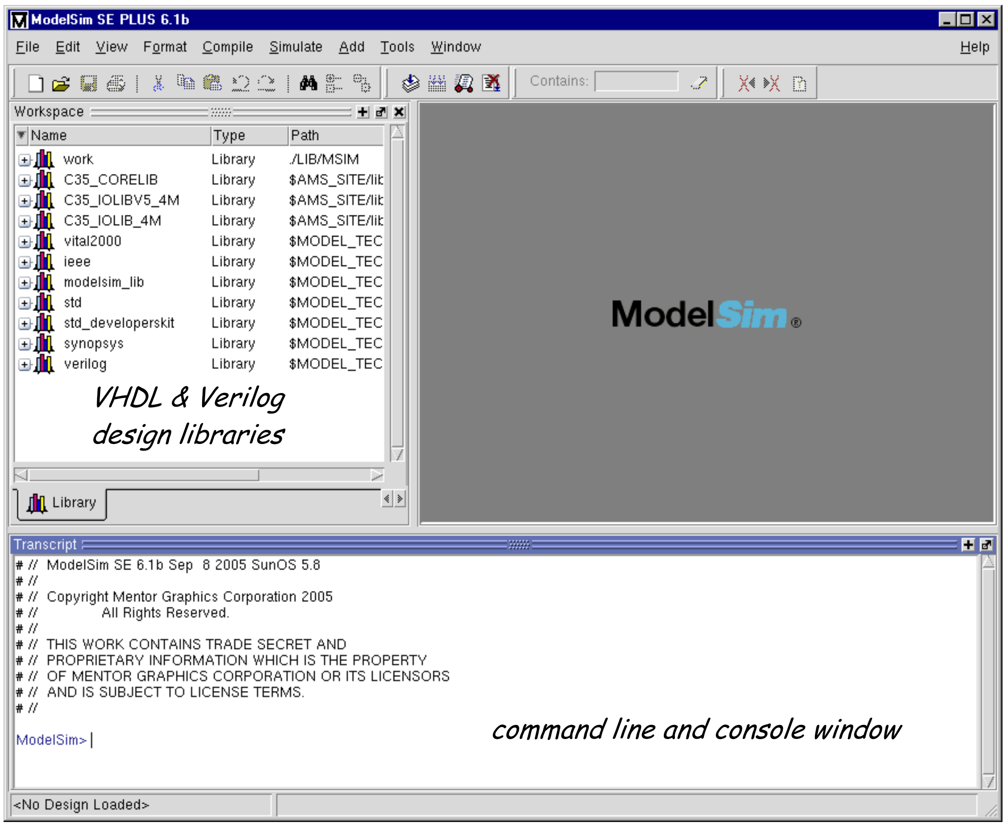
Chapter 2: VHDL and Verilog simulation

This chapter presents the main steps to perform the logic simulation of VHDL and Verilog models with the Modelsim tool. The command msim_doc gives access to the Modelsim on-line documentation.

2.1 Starting the Modelsim graphical environment

To start the Modelsim environment, enter in the **vsim** command in the Unix shell:

```
[52]vachoux@lsmsun1-ADDSUB> vsim &
```


ModelSim SE PLUS 6.1b
File Edit View Format Compile Simulate Add Tools Window Help
Contains:
Workspace
Library
ModelSim
VHDL & Verilog design libraries
Transcript
command line and console window

Name	Type	Path
work	Library	./LIB/MSIM
C35_CORELIB	Library	$AMS_SITE/lib
C35_IOLIBV5_4M	Library	$AMS_SITE/lib
C35_IOLIB_4M	Library	$AMS_SITE/lib
vital2000	Library	$MODEL_TEC
ieee	Library	$MODEL_TEC
modelsim_lib	Library	$MODEL_TEC
std	Library	$MODEL_TEC
std_developerskit	Library	$MODEL_TEC
synopsys	Library	$MODEL_TEC
verilog	Library	$MODEL_TEC

```
# //  ModelSim SE 6.1b Sep  8 2005 SunOS 5.8
# //
# //  Copyright Mentor Graphics Corporation 2005
# //              All Rights Reserved.
# //
# //  THIS WORK CONTAINS TRADE SECRET AND
# //  PROPRIETARY INFORMATION WHICH IS THE PROPERTY
# //  OF MENTOR GRAPHICS CORPORATION OR ITS LICENSORS
# //  AND IS SUBJECT TO LICENSE TERMS.
# //

ModelSim>
```

<No Design Loaded>

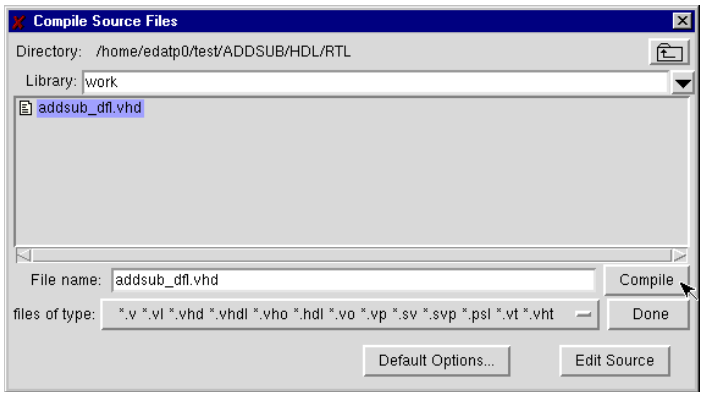
The **modelsim.ini** file actually defines the mapping between logical design libraries and their physical locations. Note that the Help menu on the top right allows one to access the complete documentation of the tool.

2.2 Simulation of (pre-synthesis) RTL VHDL models

The task here is to validate the functionality of the VHDL model that will be synthesized. The first step is to compile the VHDL model and its associated testbench. There are two ways to compile VHDL models. One way is to execute the **vcom** command from the command line of the Modelsim window:

```
ModelSim> vcom HDL/RTL/addsub_dfl.vhd
# Model Technology ModelSim SE vcom 6.1b Compiler 2005.09 Sep  8 2005
# -- Loading package standard
# -- Loading package std_logic_1164
# -- Loading package numeric_std
# -- Compiling entity addsub
# -- Compiling architecture dfl of addsub

ModelSim> vcom HDL/TBENCH/tb_addsub_rtl.vhd
# Model Technology ModelSim SE vcom 6.1b Compiler 2005.09 Sep  8 2005
# -- Loading package standard
# -- Loading package std_logic_1164
# -- Loading package numeric_std
# -- Compiling entity tb_addsub_rtl
# -- Compiling architecture bench of tb_addsub_rtl
# -- Loading entity addsub
```

The other way is to left-click on the **Compile** icon , to select the files to compile in the HDL/RTL and HDL/TBENCH directories, click on Compile and finally close the window (click **Done**).



Directory: /home/edatp0/test/ADDSUB/HDL/RTL

Library: work

addsub_dfl.vhd

File name: addsub_dfl.vhd

files of type: *.v *.vl *.vhd *.vhdl *.vho *.hdl *.vo *.vp *.sv *.svp *.psl *.vt *.vht

The compiled modules are stored in the logical library WORK which is mapped to the physical location LIB/MSIM.

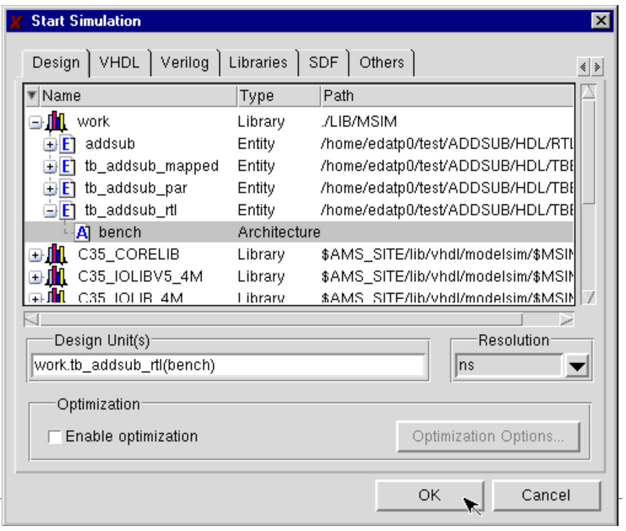
Once VHDL (or Verilog) models have been succesfully compiled in the design library, it is possible to create a make file that can be used to recompile only the required files. The **vmake** command can only be run from a Unix shell and creates the make file:

```
[53]vachoux@lsmsun1-ADDSUB> vmake > Makefile
```

The created file Makefile now defines the design unit dependencies and the compilation commands to recompile only those source files that have been modified or that depend on modified files. To rebuild the library, run the **make** command in the Unix shell (Note: in the context of the immsunsrv1 and immsunsrv2 machines, you rather have to run the command: **mgc make**).

To simulate the RTL model, select the main menu item **Simulate -> Start simulation...** to get the simulation dialog window.

Select the architecture of the testbench and a resolution of ns. Then click **OK**.



Name	Type	Path
work	Library	./LIB/MSIM
addsub	Entity	/home/edatp0/test/ADDSUB/HDL/RTI
tb_addsub_mapped	Entity	/home/edatp0/test/ADDSUB/HDL/TBI
tb_addsub_par	Entity	/home/edatp0/test/ADDSUB/HDL/TBI
tb_addsub_rtl	Entity	/home/edatp0/test/ADDSUB/HDL/TBI
bench	Architecture	
C35_CORELIB	Library	$AMS_SITE/lib/vhdl/modelsim/$MSIN
C35_IOLIBV5_4M	Library	$AMS_SITE/lib/vhdl/modelsim/$MSIN
C35_IOLIB_4M	Library	$AMS_SITE/lib/vhdl/modelsim/$MSIN

Design Unit(s): work.tb_addsub_rtl(bench)

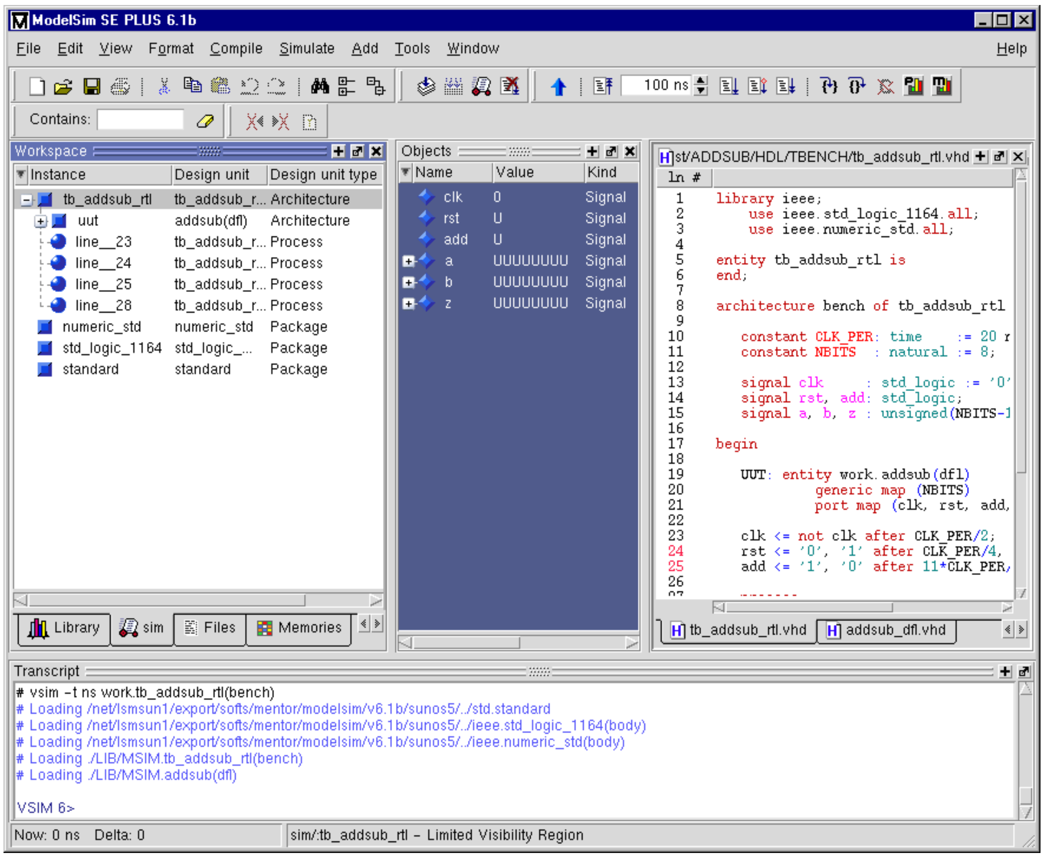
Resolution: ns

Optimization

Enable optimization

The main window now changes a bit to show the simulation hierarchy, the list of signals in the testbench and the simulation console (with now the VSIM *number>* prompt).

Left clicking twice on an instance in the simulation hierarchy pane displays the corresponding VHDL source in the right pane.


ModelSim SE PLUS 6.1b
File Edit View Format Compile Simulate Add Tools Window Help
100 ns
Contains:
Workspace
Library sim Files Memories
Objects
st/ADDSUB/HDL/TBENCH/tb_addsub_rtl.vhd
tb_addsub_rtl.vhd addsub_dfl.vhd
Transcript

Instance	Design unit	Design unit type
tb_addsub_rtl	tb_addsub_r...	Architecture
uut	addsub(dfl)	Architecture
line__23	tb_addsub_r...	Process
line__24	tb_addsub_r...	Process
line__25	tb_addsub_r...	Process
line__28	tb_addsub_r...	Process
numeric_std	numeric_std	Package
std_logic_1164	std_logic_...	Package
standard	standard	Package

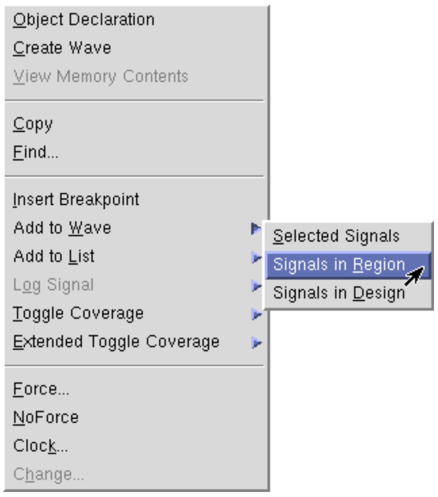
Name	Value	Kind
clk	0	Signal
rst	U	Signal
add	U	Signal
a	UUUUUUUU	Signal
b	UUUUUUUU	Signal
z	UUUUUUUU	Signal

```
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity tb_addsub_rtl is
end;

architecture bench of tb_addsub_rtl

   constant CLK_PER: time     := 20 r
   constant NBITS   : natural := 8;

   signal clk      : std_logic := '0'
   signal rst, add: std_logic;
   signal a, b, z : unsigned(NBITS-1

begin

   UUT: entity work.addsub(dfl)
            generic map (NBITS)
            port map (clk, rst, add,

   clk <= not clk after CLK_PER/2;
   rst <= '0', '1' after CLK_PER/4,
   add <= '1', '0' after 11*CLK_PER,
```

```
# vsim -t ns work.tb_addsub_rtl(bench)
# Loading /net/lsmsun1/export/softs/mentor/modelsim/v6.1b/sunos5/../std.standard
# Loading /net/lsmsun1/export/softs/mentor/modelsim/v6.1b/sunos5/../ieee.std_logic_1164(body)
# Loading /net/lsmsun1/export/softs/mentor/modelsim/v6.1b/sunos5/../ieee.numeric_std(body)
# Loading ./LIB/MSIM.tb_addsub_rtl(bench)
# Loading ./LIB/MSIM.addsub(dfl)

VSIM 6>
```

Now: 0 ns Delta: 0 | sim/:tb_addsub_rtl - Limited Visibility Region

The next step is to select the signals to display in simulation. Right click in the Objects (top center) pane, then select
Add to Wave -> Signal in Region.



Note that the appropriate hierarchy level is selected in the simulation hierarchy window. Selecting another level, e.g. uut, will display all the signals visible in this scope. You may want to add selected signals from inner levels (local signals).

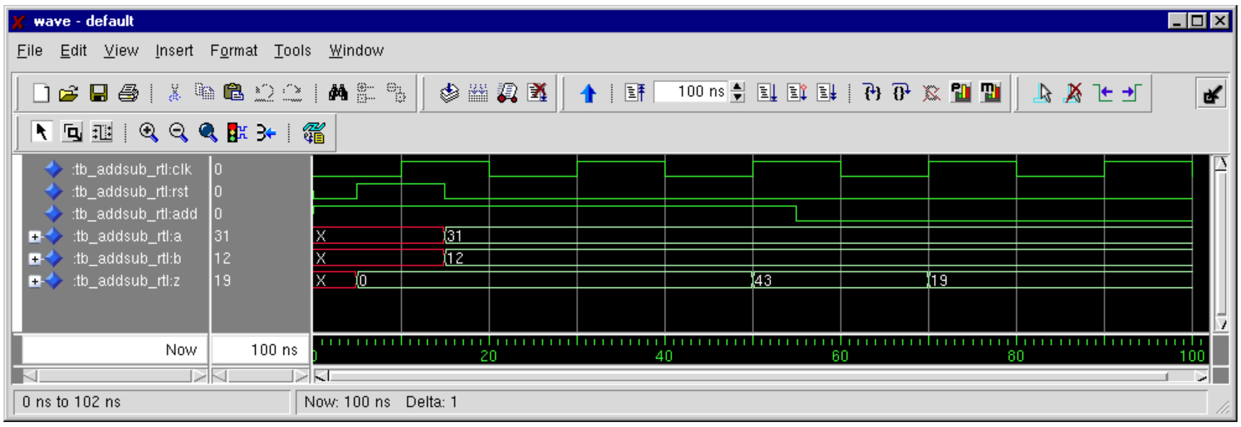
The selected signals are displayed in a new window called wave. The wave pane is by default located on the top right (as a new tab on the source windows). You can click on the **Undock** icon to make the wave pane separate.

To start the simulation, it is either possible to enter **run** commands in the simulation console such as:

```
VSIM 7> run 100 ns
```

or to click on the **Run** icon 100 ns in the main window or in the wave window.

The signal waveforms are then visible in the wave window. To change the radix of the displayed signals, select the signals (press shift left-click for multiple selection), then select the wave menu item **Format -> Radix -> Unsigned**.



Note that the command **run -all** runs the simulation until there is no more pending event in the simulation queue. This could lead to never ending simulation when the model, like the testbench loaded here, has a continuously switching signal such as the clock signal clk. It is however possible to stop the current simulation by clocking the **Break** icon in the main window or in the wave window.

If you make any modification to the VHDL source, you need to *recompile* the sources (manually or using the **vmake** command described earlier in this section), and then restart the simulation in the *same* environment (e.g., the same displayed waveforms or the same simulation breakpoints) with the **restart -f** command.

2.3 Simulation of the post-synthesis VHDL model with timing data

This step occurs after the RTL model has been synthesized into a gate-level netlist. The timing information about the design is stored in a SDF file. See "3.8 VHDL/Verilog gate-level netlist generation and post-synthesis timing data (SDF) extraction".

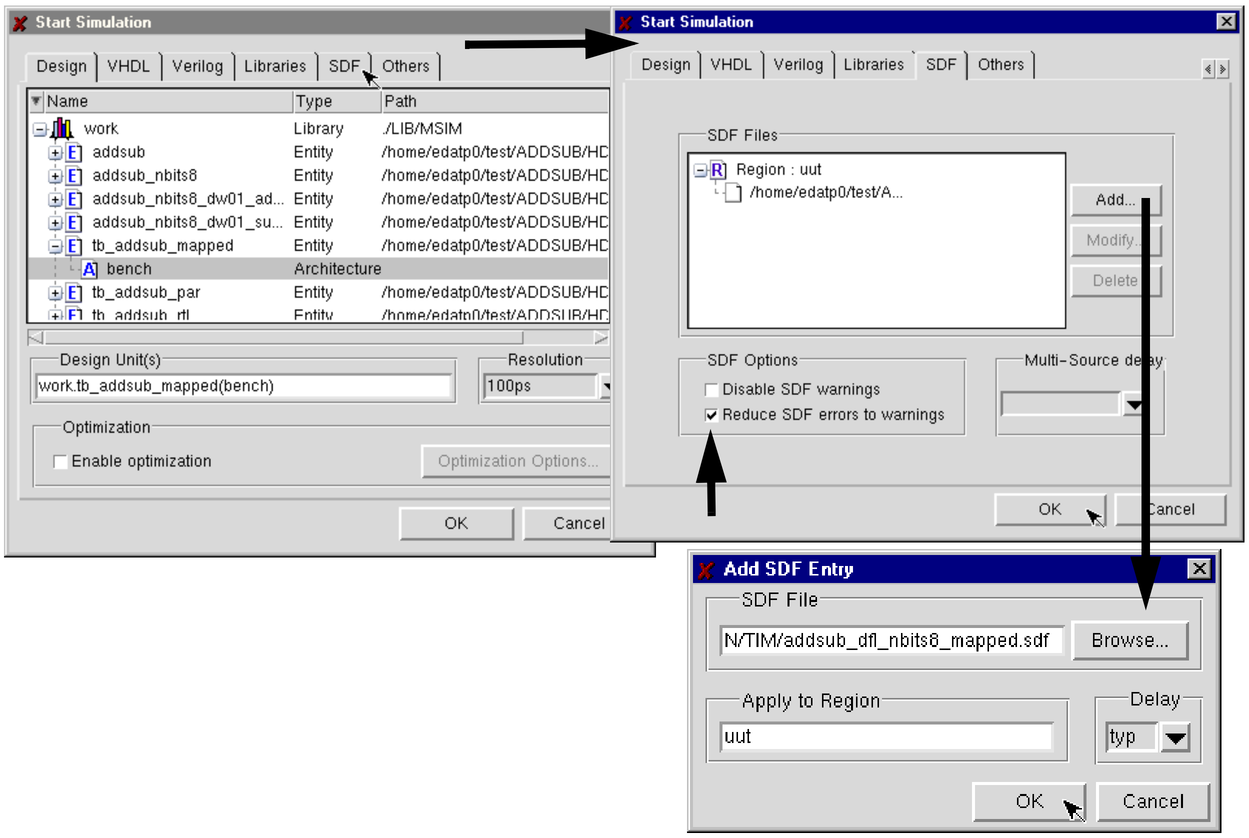
Compile the VHDL gate-level netlist generated by the logic synthesis and its testbench:

```
ModelSim> vcom HDL/GATE/addsub_dfl_nbits8_mapped.vhdl
...
ModelSim> vcom HDL/TBENCH/tb_addsub_mapped.vhd
...
```

To simulate the RTL model, select the main menu item **Simulate -> Start simulation...** to get the simulation dialog window. Select the architecture of the testbench and a resolution of 100ps. Then click the **SDF** tab.

In the SDF dialog window, add the file SYN/TIM/addsub_dfl_nbits8_mapped.sdf and specify the region UUT, which is the label of the instance in the testbench that will be annotated with timing data.

Note that the **Reduce SDF errors to warnings** box must be checked. This is required to avoid the simulation to stop prematurely due to errors such as *"Failed to find port 'a(7)'"*. These are not really errors here as they are related to interconnect delay data in the SDF file that are not used in the simulation (they are actually all set to zero).



Name	Type	Path
work	Library	./LIB/MSIM
addsub	Entity	/home/edatp0/test/ADDSUB/HD
addsub_nbits8	Entity	/home/edatp0/test/ADDSUB/HD
addsub_nbits8_dw01_ad...	Entity	/home/edatp0/test/ADDSUB/HD
addsub_nbits8_dw01_su...	Entity	/home/edatp0/test/ADDSUB/HD
tb_addsub_mapped	Entity	/home/edatp0/test/ADDSUB/HD
bench	Architecture	
tb_addsub_par	Entity	/home/edatp0/test/ADDSUB/HD
tb_addsub_rtl	Entity	/home/edatp0/test/ADDSUB/HD

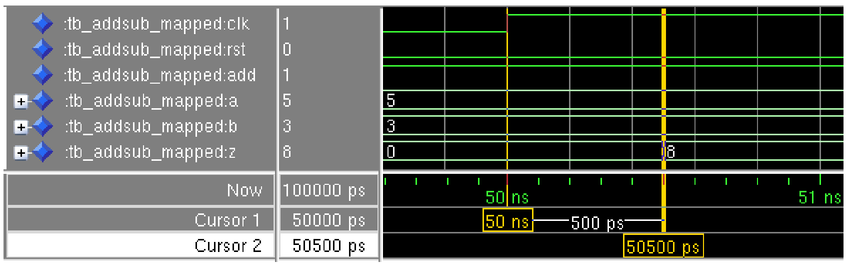
Design Unit(s): work.tb_addsub_mapped(bench)
Resolution: 100ps
Optimization: ☐ Enable optimization

SDF Files:
Region : uut
/home/edatp0/test/A...

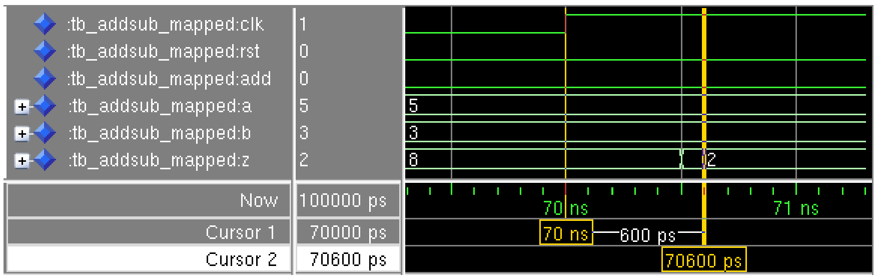
SDF Options:
☐ Disable SDF warnings
☑ Reduce SDF errors to warnings

SDF File: N/TIM/addsub_dfl_nbits8_mapped.sdf
Apply to Region: uut
Delay: typ

Then click **OK** in the remaining **Start Simulation** dialog box to load the mapped netlist. Clock to output delays of the order of 500ps to 1ns should be visible in the wave window.



:tb_addsub_mapped:clk	1
:tb_addsub_mapped:rst	0
:tb_addsub_mapped:add	1
:tb_addsub_mapped:a	5
:tb_addsub_mapped:b	3
:tb_addsub_mapped:z	8
Now	100000 ps
Cursor 1	50000 ps
Cursor 2	50500 ps



:tb_addsub_mapped:clk	1
:tb_addsub_mapped:rst	0
:tb_addsub_mapped:add	0
:tb_addsub_mapped:a	5
:tb_addsub_mapped:b	3
:tb_addsub_mapped:z	2
Now	100000 ps
Cursor 1	70000 ps
Cursor 2	70600 ps

2.4 Simulation of the post-route Verilog model with timing data

This step occurs after the design has been placed and routed. See "4.16 Post-route timing data extraction" and "4.17 Post-route netlist generation". This steps involves the simulation of a Verilog gate-level netlist with a VHDL testbench.

Before compiling the Verilog netlist and the associated VHDL testbench, it is necessary to prepare the design library. The **vdir -r** command gives the content of the design library:

```
[120]vachoux@lsmsun1-ADDSUB> vdir -r
Library Vendor : Model Technology
Maximum Unnamed Designs : 3
ENTITY addsub
  ARCH dfl
ENTITY addsub_nbits8
  ARCH syn_dfl
ENTITY addsub_nbits8_dw01_addsub_0
  ARCH syn_rpl
ENTITY tb_addsub_mapped
  ARCH bench
ENTITY tb_addsub_par
  ARCH bench
ENTITY tb_addsub_rtl
  ARCH bench
```

The ENTITY and ARCHITECTURE keywords denote VHDL design units. The problem is that the modules in the Verilog netlist have the same names as in the VHDL post-synthesis models (actually they only differ in the case of some name parts). To allow the VHDL testbench HDL/tb_addsub_par.vhd to instantiate a Verilog model, the direct instantiation statement cannot include the architecture name. Therefore, a default mapping is considered and the design library must only contain the required compiled units. As a consequence, some existing units must be deleted from the library with the **vdel** command:

```
[121]vachoux@lsmsun1-ADDSUB> vdel addsub_nbits8_dw01_addsub_0
[122]vachoux@lsmsun1-ADDSUB> vdel addsub_nbits8
```

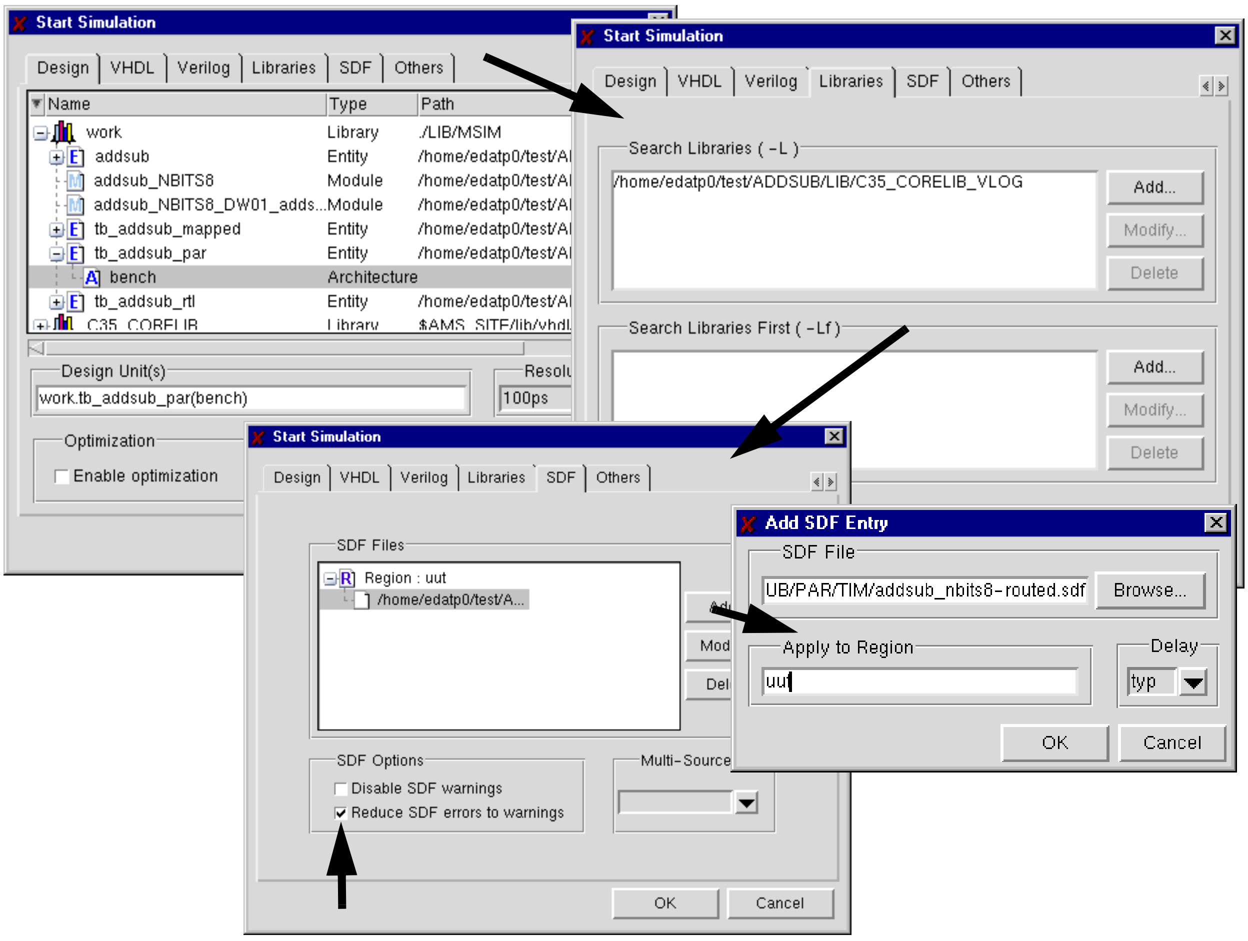
Note that this could also be done from within the Modelsim GUI.

Now you can compile the Verilog netlist and the VHDL testbench:

```
[123]vachoux@lsmsun1-ADDSUB> vlog HDL/GATE/addsub_nbits8-routed.v
[124]vachoux@lsmsun1-ADDSUB> vcom HDL/TBENCH/tb_addsub_par.vhd
```

To simulate the placed and routed netlist with timing data, select the item **Simulate -> Start simulation...** in the main menu to get the simulation dialog window. Select the architecture of the testbench and a resolution of 100ps. Then click the **Libraries** tab to add the Verilog gate library LIB/C35_CORELIB_VLOG, and load the SDF timing file PAR/SDFTIM/addsub_nbits8-routed.sdf.

Note that the **Reduce SDF errors to warnings** box must be checked. This is required to avoid the simulation to stop prematurely due to errors such as *"Failed to find matching specify timing constraint"*. These are not really errors here as they are related to removal (asynchronous) timing constraints generated by Encounter that are not supported in the Verilog models of the gates.


Start Simulation — Design

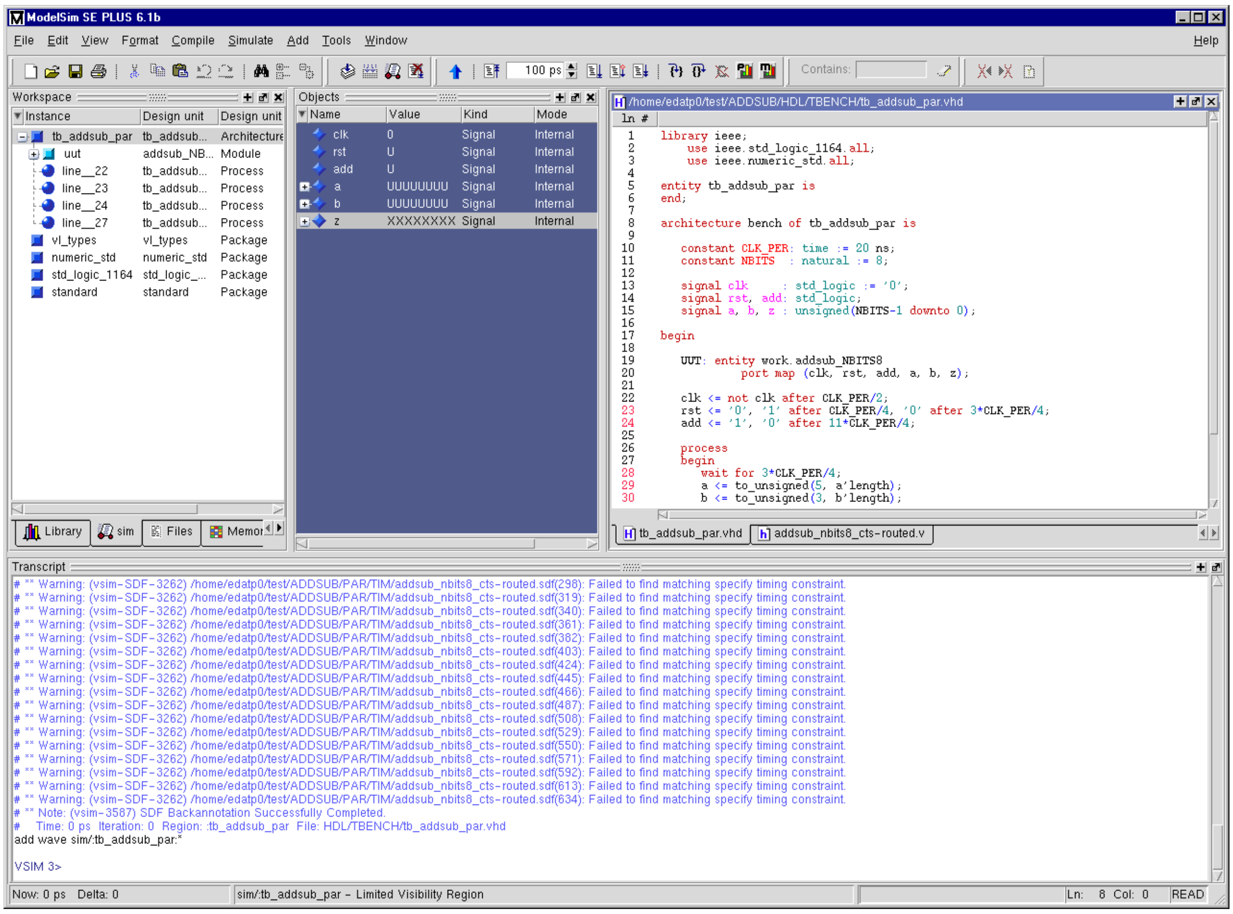
Name	Type	Path
work	Library	./LIB/MSIM
addsub	Entity	/home/edatp0/test/A
addsub_NBITS8	Module	/home/edatp0/test/A
addsub_NBITS8_DW01_adds...	Module	/home/edatp0/test/A
tb_addsub_mapped	Entity	/home/edatp0/test/A
tb_addsub_par	Entity	/home/edatp0/test/A
bench	Architecture	
tb_addsub_rtl	Entity	/home/edatp0/test/A
C35_CORELIB	Library	$AMS_SITE/lib/vhdl

Design Unit(s): work.tb_addsub_par(bench)
Resolution: 100ps
Optimization: Enable optimization [ ]

Start Simulation — Libraries
Search Libraries (-L): /home/edatp0/test/ADDSUB/LIB/C35_CORELIB_VLOG
Search Libraries First (-Lf):

Start Simulation — SDF
SDF Files: Region : uut — /home/edatp0/test/A...
SDF Options: Disable SDF warnings [ ]; Reduce SDF errors to warnings [✓]
Multi-Source

Add SDF Entry
SDF File: UB/PAR/TIM/addsub_nbits8-routed.sdf
Apply to Region: uut
Delay: typ



ModelSim SE PLUS 6.1b
File Edit View Format Compile Simulate Add Tools Window Help
100 ps
Contains:
Library
sim
Files
Memory

Workspace

Instance	Design unit	Design unit
tb_addsub_par	tb_addsub...	Architecture
uut	addsub_NB...	Module
line__22	tb_addsub...	Process
line__23	tb_addsub...	Process
line__24	tb_addsub...	Process
line__27	tb_addsub...	Process
vl_types	vl_types	Package
numeric_std	numeric_std	Package
std_logic_1164	std_logic_...	Package
standard	standard	Package

Objects

Name	Value	Kind	Mode
clk	0	Signal	Internal
rst	U	Signal	Internal
add	U	Signal	Internal
a	UUUUUUUU	Signal	Internal
b	UUUUUUUU	Signal	Internal
z	XXXXXXXX	Signal	Internal

/home/edatp0/test/ADDSUB/HDL/TBENCH/tb_addsub_par.vhd

```vhdl
library ieee;
    use ieee.std_logic_1164.all;
    use ieee.numeric_std.all;

entity tb_addsub_par is
end;

architecture bench of tb_addsub_par is

   constant CLK_PER: time := 20 ns;
   constant NBITS   : natural := 8;

   signal clk      : std_logic := '0';
   signal rst, add: std_logic;
   signal a, b, z : unsigned(NBITS-1 downto 0);

begin

   UUT: entity work.addsub_NBITS8
           port map (clk, rst, add, a, b, z);

   clk <= not clk after CLK_PER/2;
   rst <= '0', '1' after CLK_PER/4, '0' after 3*CLK_PER/4;
   add <= '1', '0' after 11*CLK_PER/4;

   process
   begin
      wait for 3*CLK_PER/4;
      a <= to_unsigned(5, a'length);
      b <= to_unsigned(3, b'length);
```

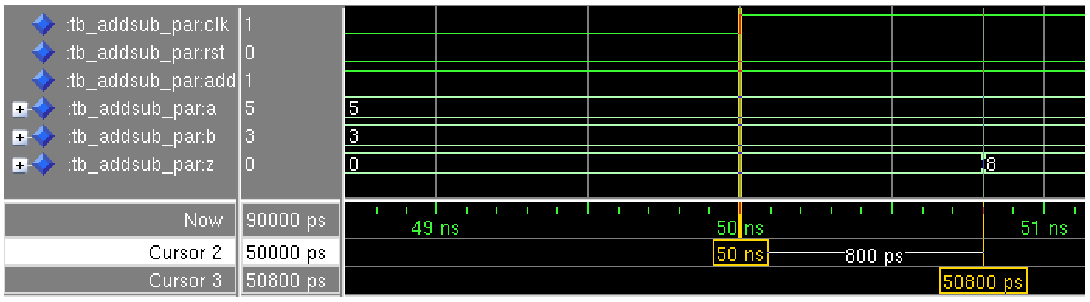
tb_addsub_par.vhd | addsub_nbits8_cts-routed.v

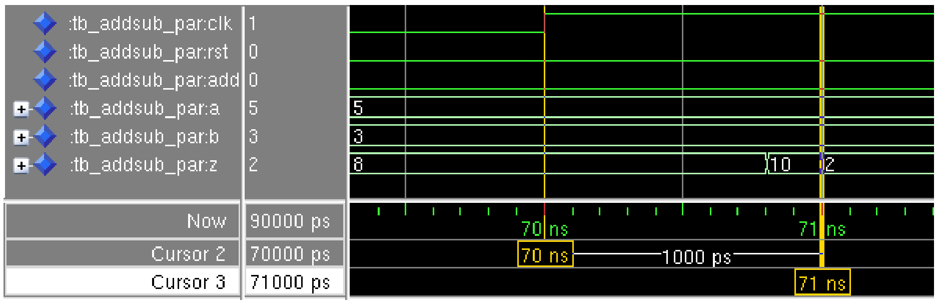
Transcript

```
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(298): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(319): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(340): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(361): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(382): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(403): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(424): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(445): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(466): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(487): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(508): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(529): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(550): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(571): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(592): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(613): Failed to find matching specify timing constraint.
# ** Warning: (vsim-SDF-3262) /home/edatp0/test/ADDSUB/PAR/TIM/addsub_nbits8_cts-routed.sdf(634): Failed to find matching specify timing constraint.
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Region: :tb_addsub_par  File: HDL/TBENCH/tb_addsub_par.vhd
add wave sim/:tb_addsub_par:*

VSIM 3>
```

Now: 0 ps Delta: 0 | sim/:tb_addsub_par – Limited Visibility Region | Ln: 8 Col: 0 | READ

Then click **OK** in the remaining **Start Simulation** dialog box to load the mapped netlist. Clock to output delays of the order of 800ps to 1000ps should be visible in the wave window.





More accurate values for the delays can be obtained using a smaller time unit in simulation (e.g., 10ps or less).

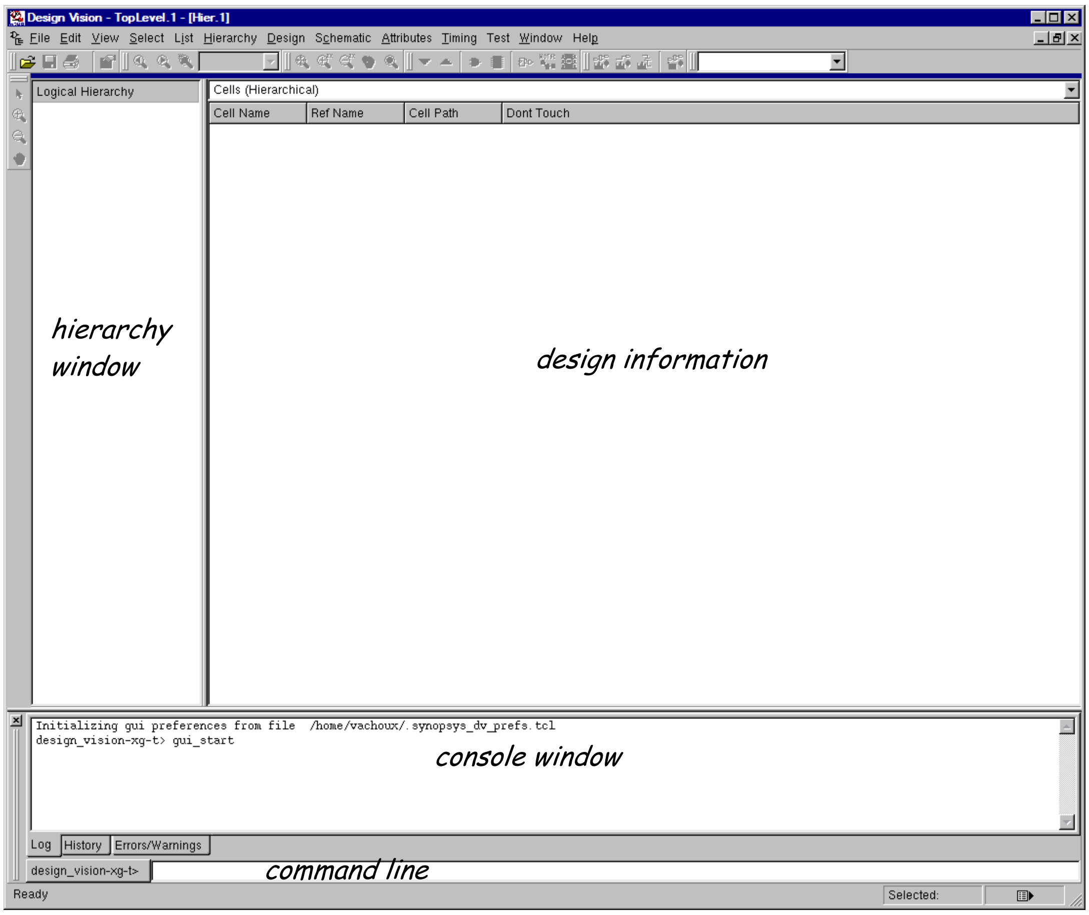
Chapter 3: Logic synthesis

This chapter presents the main steps to perform the logic synthesis of the VHDL RTL model with the Synopsys Design Vision and Design Compiler tools. The **sold** alias displays the complete Synopsys documentation set. Manual pages are available by executing the command “snps man *command*” (e.g., snps man design_vision).

3.1 Starting the Design Vision graphical environment

To start the Synopsys Design Vision environment, enter the **design_vision** command in a *new* shell:

```
[65]vachoux@lsmsun1-ADDSUB> design_vision
```



```
Initializing gui preferences from file  /home/vachoux/.synopsys_dv_prefs.tcl
design_vision-xg-t> gui_start
```

The command line is also echoed in the terminal shell from which the tool has been started, so it is possible to enter DC commands from there as well (the shell has the design_vision-xg-t> prompt). It is still possible to execute some Unix commands from here.

3.2 RTL VHDL model analysis

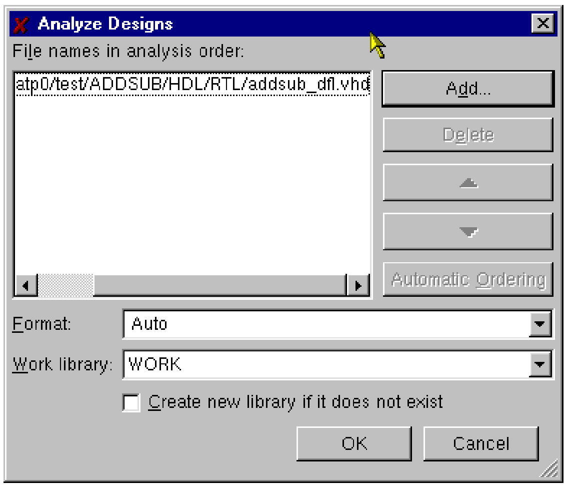
The analysis phase compiles the VHDL model and checks that the VHDL code is synthesizable.

Select **File -> Analyze...** in the main menu.

Use the **Add...** button to add all the VHDL sources you need to analyze.

In the case you have more than one VHDL file to analyze, be careful to list the files in the correct analysis order. Another way is to select all the files and click on the button **Automatic Ordering** to let the tool find the right dependency order. This button is only active when several VHDL files are listed.

Click **OK**.



atp0/test/ADDSUB/HDL/RTL/addsub_dfl.vhd

Format: Auto

Work library: WORK

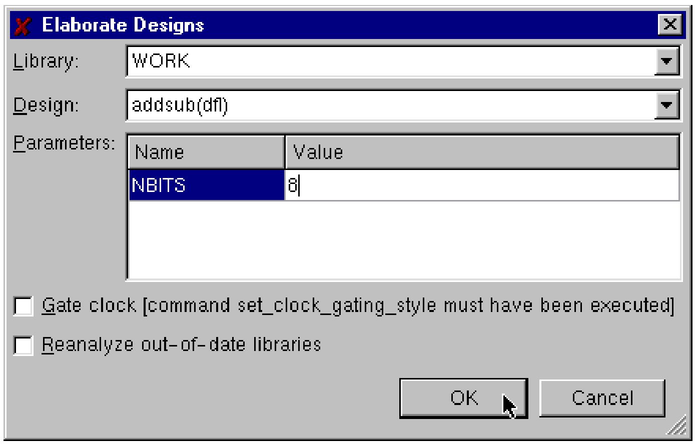
☐ Create new library if it does not exist

3.3 Design elaboration

The elaboration phase performs a generic pre-synthesis of the analyzed model. It essentially identifies the registers that will be inferred.

Select **File -> Elaborate...** in the main menu. The DEFAULT library is identical to the WORK library. Specify the value for the NBITS generic parameter to 8.

Click **OK**.



Library: WORK

Design: addsub(dfl)

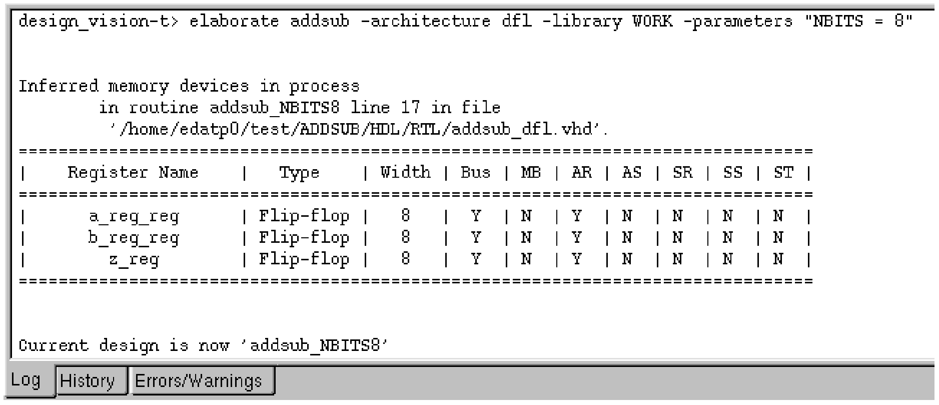
Parameters:

Name	Value
NBITS	8

☐ Gate clock [command set_clock_gating_style must have been executed]

☐ Reanalyze out-of-date libraries

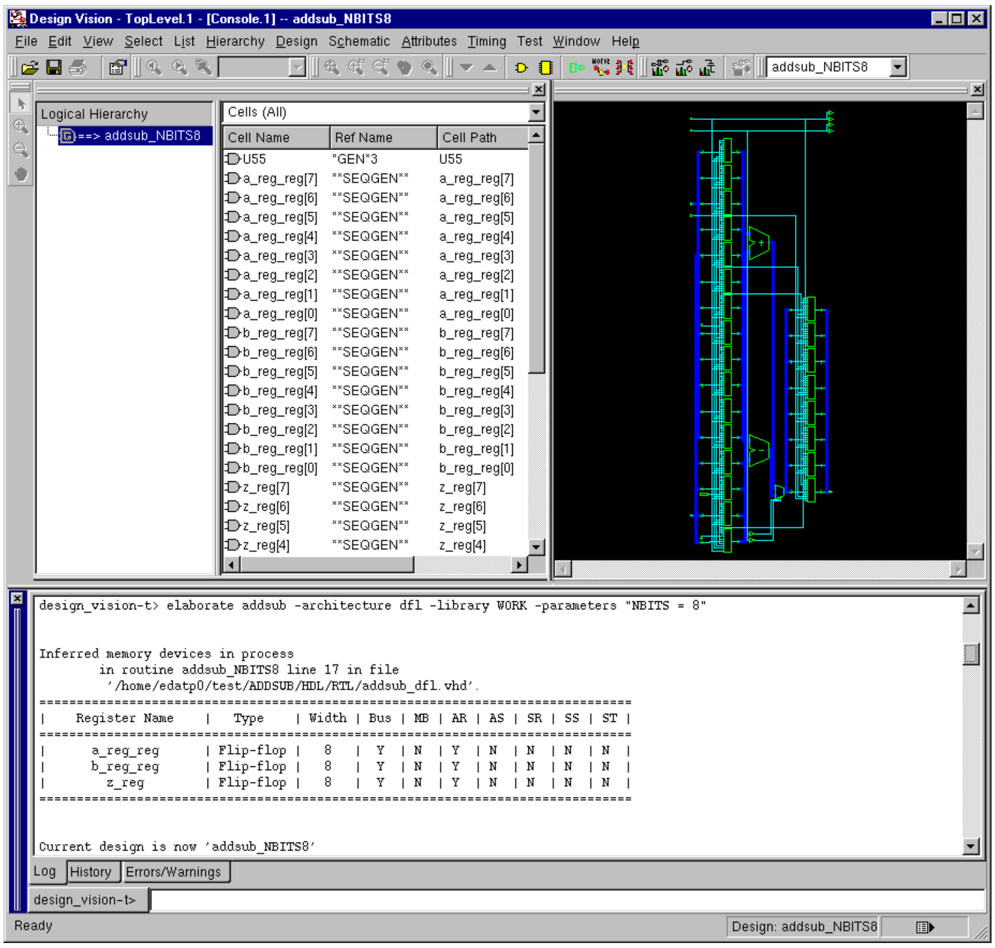
The console now displays the inferred registers and the kind of reset (here asynchronous reset - AR: Y).



```
design_vision-t> elaborate addsub -architecture dfl -library WORK -parameters "NBITS = 8"


Inferred memory devices in process
        in routine addsub_NBITS8 line 17 in file
         '/home/edatp0/test/ADDSUB/HDL/RTL/addsub_dfl.vhd'.
===============================================================================
|    Register Name     |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|       a_reg_reg      | Flip-flop |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
|       b_reg_reg      | Flip-flop |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
|         z_reg        | Flip-flop |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================


Current design is now 'addsub_NBITS8'
```

Note the name addsub_NBITS8 given to the elaborated entity.

It is possible to display the elaborated schematic by selecting the entity addsub_NBITS8 in the hierarchy window and then clicking the **Create Design Schematic** icon . Note that the symbols merely indicate generic components that do not yet represent any real logic gate.


Design Vision - TopLevel.1 - [Console.1] -- addsub_NBITS8
File Edit View Select List Hierarchy Design Schematic Attributes Timing Test Window Help
addsub_NBITS8
Logical Hierarchy
==> addsub_NBITS8
Cells (All)
Log History Errors/Warnings
design_vision-t>
Ready
Design: addsub_NBITS8

Cell Name	Ref Name	Cell Path
U55	*GEN*3	U55
a_reg_reg[7]	**SEQGEN**	a_reg_reg[7]
a_reg_reg[6]	**SEQGEN**	a_reg_reg[6]
a_reg_reg[5]	**SEQGEN**	a_reg_reg[5]
a_reg_reg[4]	**SEQGEN**	a_reg_reg[4]
a_reg_reg[3]	**SEQGEN**	a_reg_reg[3]
a_reg_reg[2]	**SEQGEN**	a_reg_reg[2]
a_reg_reg[1]	**SEQGEN**	a_reg_reg[1]
a_reg_reg[0]	**SEQGEN**	a_reg_reg[0]
b_reg_reg[7]	**SEQGEN**	b_reg_reg[7]
b_reg_reg[6]	**SEQGEN**	b_reg_reg[6]
b_reg_reg[5]	**SEQGEN**	b_reg_reg[5]
b_reg_reg[4]	**SEQGEN**	b_reg_reg[4]
b_reg_reg[3]	**SEQGEN**	b_reg_reg[3]
b_reg_reg[2]	**SEQGEN**	b_reg_reg[2]
b_reg_reg[1]	**SEQGEN**	b_reg_reg[1]
b_reg_reg[0]	**SEQGEN**	b_reg_reg[0]
z_reg[7]	**SEQGEN**	z_reg[7]
z_reg[6]	**SEQGEN**	z_reg[6]
z_reg[5]	**SEQGEN**	z_reg[5]
z_reg[4]	**SEQGEN**	z_reg[4]

```
design_vision-t> elaborate addsub -architecture dfl -library WORK -parameters "NBITS = 8"


Inferred memory devices in process
        in routine addsub_NBITS8 line 17 in file
         '/home/edatp0/test/ADDSUB/HDL/RTL/addsub_dfl.vhd'.
===============================================================================
|    Register Name     |   Type    | Width | Bus | MB | AR | AS | SR | SS | ST |
===============================================================================
|      a_reg_reg       | Flip-flop |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
|      b_reg_reg       | Flip-flop |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
|        z_reg         | Flip-flop |   8   |  Y  | N  | Y  | N  | N  | N  | N  |
===============================================================================


Current design is now 'addsub_NBITS8'
```

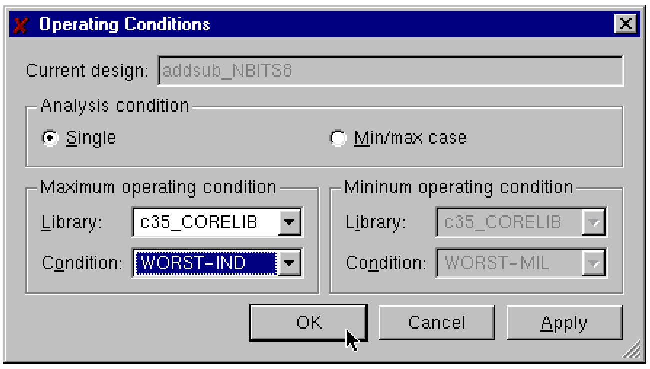
3.4 Design environment definition

Before a design can be optimized, you must define the environment in which the design is expected to operate. You define the environment by specifying operating conditions, wire load models, and system interface characteristics. Operating conditions include temperature, voltage, and process variations. Wire load models estimate the effect of wire length on design performance. System interface characteristics include input drives, input and output loads, and fanout loads. The environment model directly affects design synthesis results. Here we will only deal with operating conditions and wire load models.

To define the operating conditions, select the main menu item
Attributes -> Operating Environment -> Operating Conditions...

Select the WORST-IND condition, which defines a temperature of 85°C, a voltage of 3V (the C35 process is a 3.3V process), and a slow process. Each cell library defines its own set of operating conditions and may use different names for each set.

Click **OK**.



Current design: addsub_NBITS8

Analysis condition: [✓ Single] Min/max case

Maximum operating condition — Library: c35_CORELIB — Condition: WORST-IND

Mininum operating condition — Library: c35_CORELIB — Condition: WORST-MIL

Wire load models allows the tool to estimate the effect of wire length and fanout on the resistance, capacitance, and area of nets. The AMS design kit defines a number of wire load models. It also defines an automatic selection of the wire load model to use according to the design area, which is actually considered here.

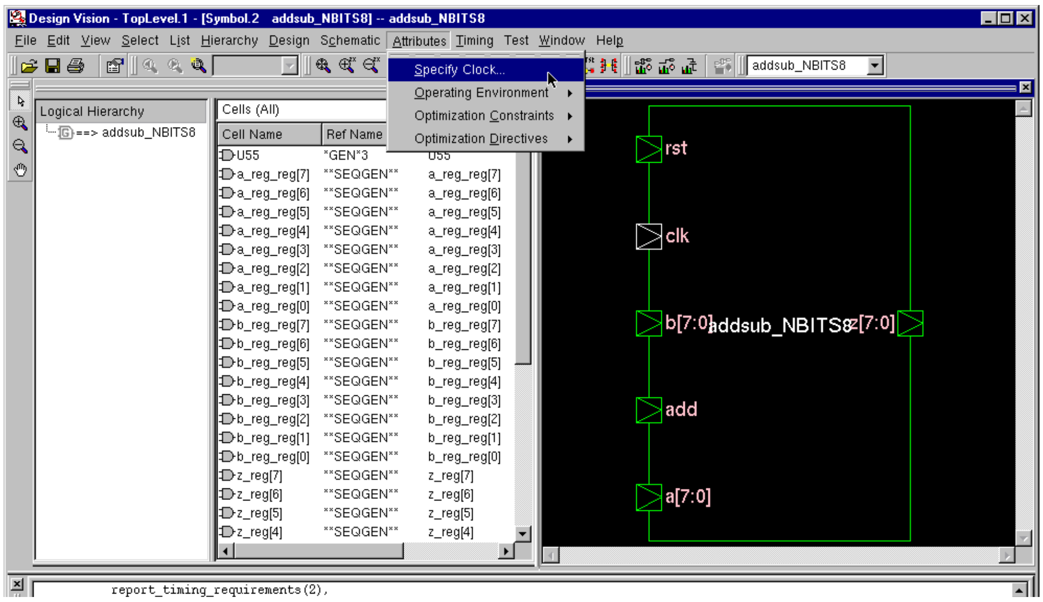
To get the definitions of the available operating conditions (and on the cell library), execute the **report_lib** command in the tool command line:

```
report_lib c35_CORELIB
```

The **report_design** command summarizes the definitions of the design environment.

3.5 Design constraint definitions

Many kinds of constraints may be defined on the design. Here only constraints on the area and the clock will be defined. To define the clock attributes, i.e. its period and duty cycle, select the entity addsub_NBITS8 in the hierarchy window and then click the **Create Symbol View** icon .



Cell Name	Ref Name	
U55	*GEN*3	U55
a_reg_reg[7]	**SEQGEN**	a_reg_reg[7]
a_reg_reg[6]	**SEQGEN**	a_reg_reg[6]
a_reg_reg[5]	**SEQGEN**	a_reg_reg[5]
a_reg_reg[4]	**SEQGEN**	a_reg_reg[4]
a_reg_reg[3]	**SEQGEN**	a_reg_reg[3]
a_reg_reg[2]	**SEQGEN**	a_reg_reg[2]
a_reg_reg[1]	**SEQGEN**	a_reg_reg[1]
a_reg_reg[0]	**SEQGEN**	a_reg_reg[0]
b_reg_reg[7]	**SEQGEN**	b_reg_reg[7]
b_reg_reg[6]	**SEQGEN**	b_reg_reg[6]
b_reg_reg[5]	**SEQGEN**	b_reg_reg[5]
b_reg_reg[4]	**SEQGEN**	b_reg_reg[4]
b_reg_reg[3]	**SEQGEN**	b_reg_reg[3]
b_reg_reg[2]	**SEQGEN**	b_reg_reg[2]
b_reg_reg[1]	**SEQGEN**	b_reg_reg[1]
b_reg_reg[0]	**SEQGEN**	b_reg_reg[0]
z_reg[7]	**SEQGEN**	z_reg[7]
z_reg[6]	**SEQGEN**	z_reg[6]
z_reg[5]	**SEQGEN**	z_reg[5]
z_reg[4]	**SEQGEN**	z_reg[4]

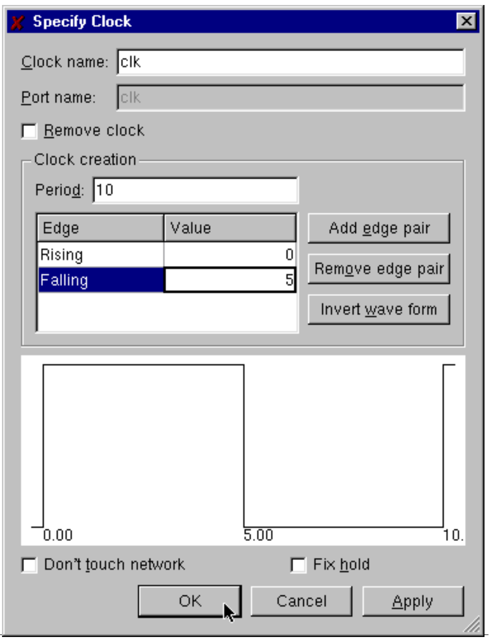
```
report_timing_requirements(2),
```

In the symbol view, select the clk pin and then select the main menu item **Attributes -> Specify Clock...**.

Define a clock period of 10 ns with 50% duty cycle. Time unit is not specified here. It is defined in the cell library and is usually ns.

Click **OK**. The console now includes the command line equivalent of the clock definition:

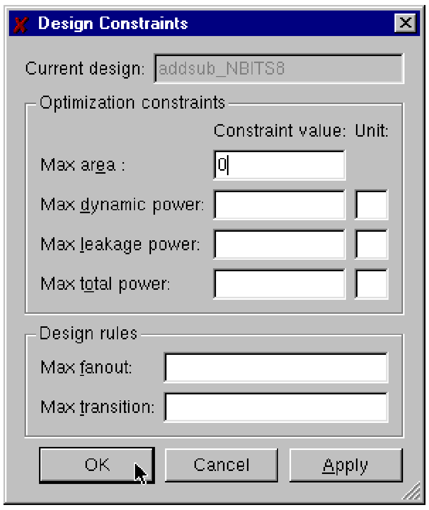
```
create_clock -name "clk" -period 10
      -waveform { 0 5  }  { clk  }
```



Clock name: clk
Port name: clk
☐ Remove clock
Period: 10

Edge	Value
Rising	0
Falling	5

☐ Don't touch network ☐ Fix hold


Design Constraints

Current design: addsub_NBITS8

Optimization constraints

	Constraint value:	Unit:
Max area :	0	
Max dynamic power:		
Max leakage power:		
Max total power:		

Design rules

Max fanout:	
Max transition:	

To define an area constraint, select in the main menu the item **Attributes -> Optimization Constraints -> Design Constraints...**.

A max area constraint set to zero is not realistic but it will force the synthesizer to target a minimum area.

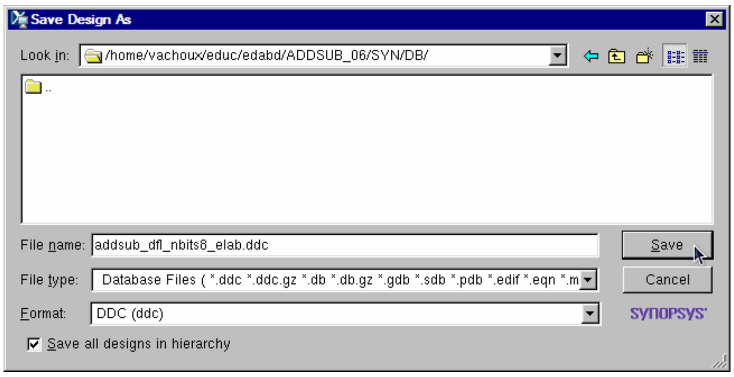
Click **OK**. The console now includes the command line equivalent of the constraint definition:

```
    set_max_area 0
```

It is a now good idea to save the elaborated design so it will be possible to run several optimization steps from that point.

Select the entity addsub_NBITS8 in the hierarchy window and then the main menu item **File -> Save As...**.

Save the elaborated design under the name addsub_dfl_nbits8_elab.ddc in the SYN/DB directory.


Save Design As

Look in: /home/vachoux/educ/edabd/ADDSUB_06/SYN/DB/

File name: addsub_dfl_nbits8_elab.ddc

File type: Database Files (*.ddc *.ddc.gz *.db *.db.gz *.gdb *.sdb *.pdb *.edif *.eqn *.m

Format: DDC (ddc)

[✓] Save all designs in hierarchy

The selection of the option *Save all design in hierarchy* is relevant for hierarchical designs.

The console includes the equivalent command line:

```
    write -hierarchy -format ddc
          -output .../ADDSUB/SYN/DB/addsub_dfl_nbits8_elab.ddc
```

To read back an elaborated design, select the main menu item **File -> Read...** and then select the file to read.

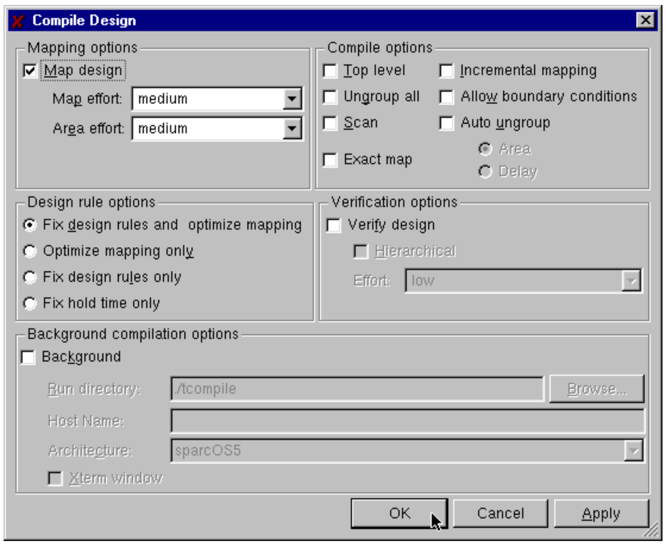
3.6 Design mapping and optimization

The optimization phase, also called here compilation phase, is technology dependent. It performs the assignment of logic gates from the standard cell library to the elaborated design in such a way the defined constraints are met.

Select the main menu item
Design -> Compile Design....

For a first run there is no need to change the default settings.

Click **OK**. The console and the Unix shell now include the progress of the work.


Compile Design

Mapping options
[✓] Map design
Map effort: medium
Area effort: medium

Compile options
[ ] Top level [ ] Incremental mapping
[ ] Ungroup all [ ] Allow boundary conditions
[ ] Scan [ ] Auto ungroup
[ ] Exact map ◉ Area ○ Delay

Design rule options
◉ Fix design rules and optimize mapping
○ Optimize mapping only
○ Fix design rules only
○ Fix hold time only

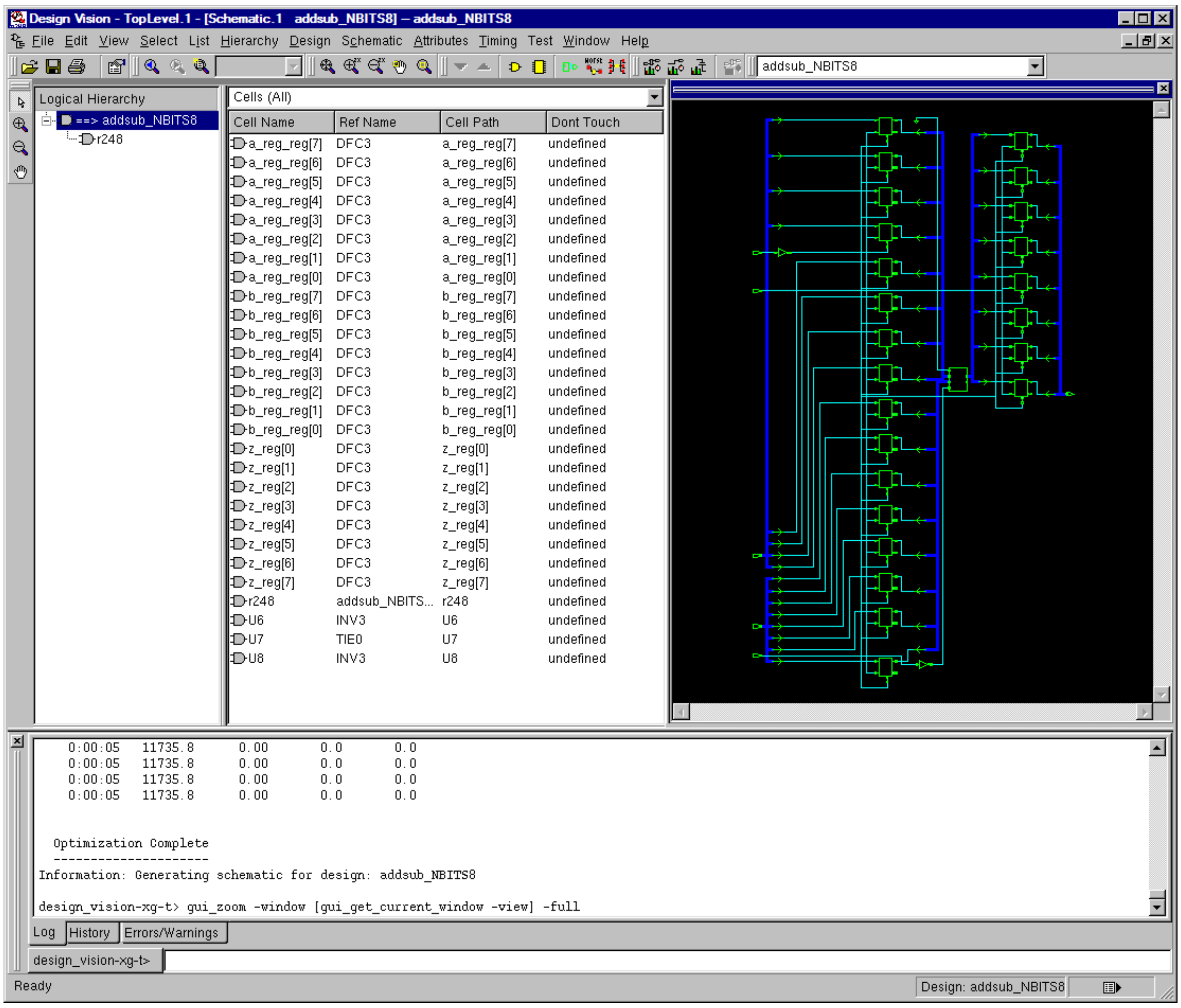
Verification options
[ ] Verify design
[ ] Hierarchical
Effort: low

Background compilation options
[ ] Background
Run directory: ./tcompile
Host Name:
Architecture: sparcOS5
[ ] Xterm window

The equivalent command line is:

```
compile -map_effort medium -area_effort medium
```

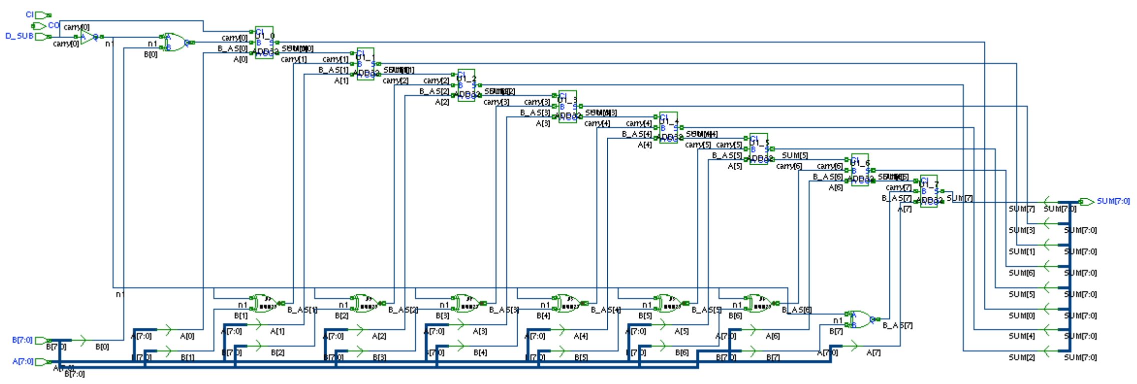
The mapped design schematics is now hierarchical as it includes one instance of a 8 bit adder and one instance of a 8 bit subtractor. Also, the cells are now real gates from the cell library.


Design Vision - TopLevel.1 - [Schematic.1 addsub_NBITS8] -- addsub_NBITS8
File Edit View Select List Hierarchy Design Schematic Attributes Timing Test Window Help
addsub_NBITS8
Logical Hierarchy
==> addsub_NBITS8
r248
Cells (All)
Log History Errors/Warnings
design_vision-xg-t>
Ready
Design: addsub_NBITS8

Cell Name	Ref Name	Cell Path	Dont Touch
a_reg_reg[7]	DFC3	a_reg_reg[7]	undefined
a_reg_reg[6]	DFC3	a_reg_reg[6]	undefined
a_reg_reg[5]	DFC3	a_reg_reg[5]	undefined
a_reg_reg[4]	DFC3	a_reg_reg[4]	undefined
a_reg_reg[3]	DFC3	a_reg_reg[3]	undefined
a_reg_reg[2]	DFC3	a_reg_reg[2]	undefined
a_reg_reg[1]	DFC3	a_reg_reg[1]	undefined
a_reg_reg[0]	DFC3	a_reg_reg[0]	undefined
b_reg_reg[7]	DFC3	b_reg_reg[7]	undefined
b_reg_reg[6]	DFC3	b_reg_reg[6]	undefined
b_reg_reg[5]	DFC3	b_reg_reg[5]	undefined
b_reg_reg[4]	DFC3	b_reg_reg[4]	undefined
b_reg_reg[3]	DFC3	b_reg_reg[3]	undefined
b_reg_reg[2]	DFC3	b_reg_reg[2]	undefined
b_reg_reg[1]	DFC3	b_reg_reg[1]	undefined
b_reg_reg[0]	DFC3	b_reg_reg[0]	undefined
z_reg[0]	DFC3	z_reg[0]	undefined
z_reg[1]	DFC3	z_reg[1]	undefined
z_reg[2]	DFC3	z_reg[2]	undefined
z_reg[3]	DFC3	z_reg[3]	undefined
z_reg[4]	DFC3	z_reg[4]	undefined
z_reg[5]	DFC3	z_reg[5]	undefined
z_reg[6]	DFC3	z_reg[6]	undefined
z_reg[7]	DFC3	z_reg[7]	undefined
r248	addsub_NBITS...	r248	undefined
U6	INV3	U6	undefined
U7	TIE0	U7	undefined
U8	INV3	U8	undefined

```
    0:00:05    11735.8     0.00      0.0      0.0
    0:00:05    11735.8     0.00      0.0      0.0
    0:00:05    11735.8     0.00      0.0      0.0
    0:00:05    11735.8     0.00      0.0      0.0


  Optimization Complete
  ---------------------
Information: Generating schematic for design: addsub_NBITS8

design_vision-xg-t> gui_zoom -window [gui_get_current_window -view] -full
```

The inferred 8 bit adder has a ripple-carry architecture and uses the available ADD32 standard cell that implements a 1 bit full adder.



The inferred 8 bit subtractor also uses a riple-carry architecture.

Note that the default resource allocation and implementation for operative parts is based on timing constraints. This means that resource sharing is used so that timing constraints are met or not worsened. In our case, a single adder has been inferred for both adder and subtractor operations.

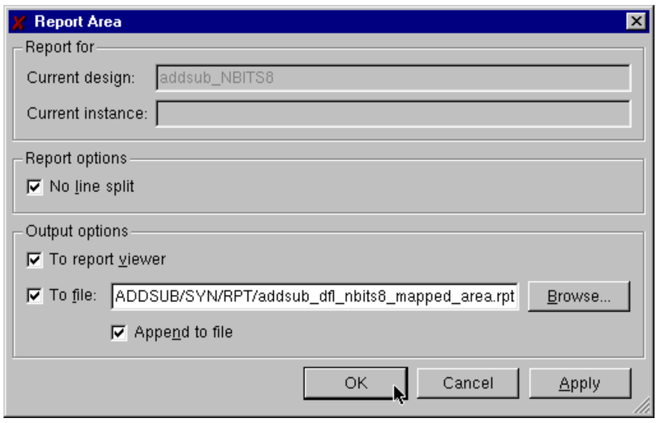
The mapped design can now be saved. Select the entity addsub_NBITS8 in the hierarchy window and then the main menu item **File -> Save As...**. Save the mapped design under the name addsub_dfl_nbits8_mapped.ddc in the directory SYN/DB.

3.7 Report generation

It is possible to get many reports on various synthesis results. Here only reports on the area used, critical path timing and the resources used will be generated.

To get a report of the area used by the mapped design, select the main menu item **Design > Report Area...**.

Save the report in the file
SYN/RPT/addsub_dfl_nbits8_mapped_area.rpt
as well as in the report viewer.



Report Area

Report for
Current design: addsub_NBITS8
Current instance:

Report options
[✓] No line split

Output options
[✓] To report viewer
[✓] To file: ADDSUB/SYN/RPT/addsub_dfl_nbits8_mapped_area.rpt
[✓] Append to file

Click **OK**. A new window and the console now display the report:

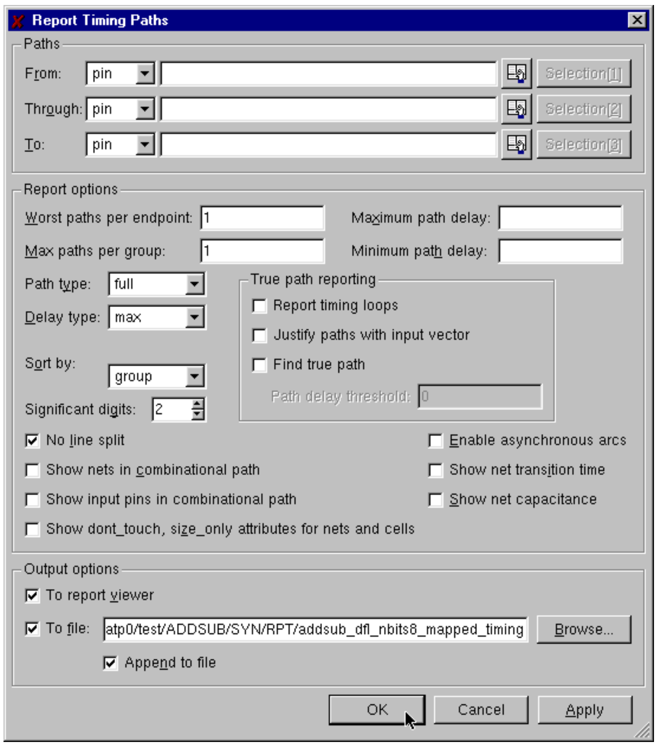
```
****************************************
Report : area
Design : addsub_NBITS8
Version: X-2005.09-SP4
Date   : Thu Nov 30 11:22:54 2006
****************************************

Library(s) Used:

    c35_CORELIB (File: /dkits/ams/hk370/synopsys/c35_3.3V/c35_CORELIB.db)

Number of ports:               27
Number of nets:                54
Number of cells:               28
Number of references:           4

Combinational area:       3203.199463
Noncombinational area:    7425.598633
Net Interconnect area:    1107.000000

Total cell area:          10628.799805
Total area:               11735.799805
```

The area unit depends on the standard cell library. Here all area figures are in square microns. The net interconnect area is estimated with the use of a wire load model that has been automatically selected from the design area.

To get a report on the most critical timing path in the mapped design, select the main menu item **Timing -> Report Timing...**.

Save the report in the file
SYN/RPT/addsub_dfl_nbits8_mapped_timing.rpt
as well as in the report viewer.

Click **OK**. A new window and the console now display the report:



Report Timing Paths

Paths
- From: pin
- Through: pin
- To: pin

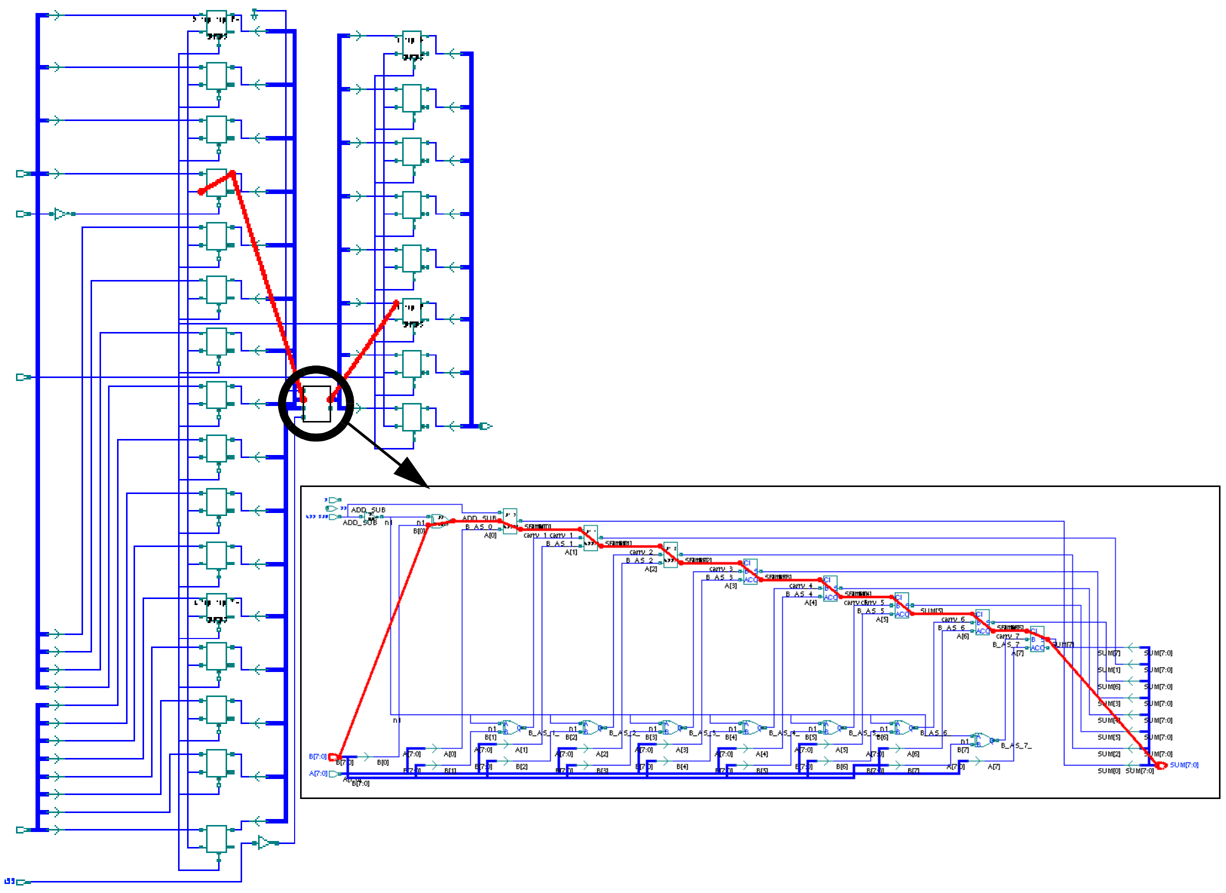
Report options
- Worst paths per endpoint: 1
- Maximum path delay:
- Max paths per group: 1
- Minimum path delay:
- Path type: full
- Delay type: max
- Sort by: group
- Significant digits: 2
- True path reporting
  - ☐ Report timing loops
  - ☐ Justify paths with input vector
  - ☐ Find true path
  - Path delay threshold: 0
- ☑ No line split
- ☐ Enable asynchronous arcs
- ☐ Show nets in combinational path
- ☐ Show net transition time
- ☐ Show input pins in combinational path
- ☐ Show net capacitance
- ☐ Show dont_touch, size_only attributes for nets and cells

Output options
- ☑ To report viewer
- ☑ To file: atp0/test/ADDSUB/SYN/RPT/addsub_dfl_nbits8_mapped_timing
- ☑ Append to file

```
    ********************************
    Report : timing
            -path full
            -delay max
            -max_paths 1
    Design : addsub_NBITS8
    Version: X-2005.09-SP4
    Date   : Thu Nov 30 11:20:15 2006
    ********************************

    Operating Conditions: WORST-IND
    Library: c35_CORELIB
    Wire Load Model Mode: enclosed

      Startpoint: b_reg_reg_0_
                  (rising edge-triggered
    flip-flop clocked by clk)
      Endpoint: z_reg_7_ (rising edge-triggered flip-flop clocked by clk)
      Path Group: clk
      Path Type: max

      Des/Clust/Port     Wire Load Model       Library
      ------------------------------------------------
      addsub_NBITS8      10k                   c35_CORELIB
      addsub_NBITS8_DW01_addsub_1
                         10k                   c35_CORELIB

      Point                                                   Incr       Path
      --------------------------------------------------------------------------
      clock clk (rise edge)                                   0.00       0.00
      clock network delay (ideal)                             0.00       0.00
      b_reg_reg_0_/C (DFC3)                                   0.00       0.00 r
      b_reg_reg_0_/Q (DFC3)                                   1.09       1.09 f
      r248/B[0] (addsub_NBITS8_DW01_addsub_1)                 0.00       1.09 f
      r248/U9/Q (XNR20)                                       1.11       2.20 f
      r248/U1_0/CO (ADD32)                                    0.99       3.20 f
      r248/U1_1/CO (ADD32)                                    0.66       3.85 f
      r248/U1_2/CO (ADD32)                                    0.66       4.51 f
      r248/U1_3/CO (ADD32)                                    0.66       5.17 f
      r248/U1_4/CO (ADD32)                                    0.66       5.82 f
      r248/U1_5/CO (ADD32)                                    0.66       6.48 f
      r248/U1_6/CO (ADD32)                                    0.66       7.13 f
      r248/U1_7/S (ADD32)                                     0.83       7.96 r
      r248/SUM[7] (addsub_NBITS8_DW01_addsub_1)               0.00       7.96 r
      z_reg_7_/D (DFC3)                                       0.00       7.96 r
      data arrival time                                                  7.96

      clock clk (rise edge)                                  10.00      10.00
      clock network delay (ideal)                             0.00      10.00
      z_reg_7_/C (DFC3)                                       0.00      10.00 r
      library setup time                                     -0.01       9.99
      data required time                                                 9.99
      --------------------------------------------------------------------------
      data required time                                                 9.99
      data arrival time                                                 -7.96
      --------------------------------------------------------------------------
      slack (MET)                                                        2.03
```

All times are expressed in ns (the time unit is defined in the cell library). The *slack* defines the time margin from the clock period. A *positive slack* means that the latest arriving signal in the path still arrives before the end of the clock period. A *negative slack* means that the timing constraint imposed by the clock is violated. The timing delays that are accounted for are the internal gate delays (from the cell library) and the estimated interconnect delays (from the cell library and the wire load model in use).

To highlight the critical path on the schematic, select the addsub_NBITS8 entity in the hierarchy window and then the **View** menu item **Highlight -> Critical Path** (CTRL-H). You can see that the path goes through the r248 component so you can also select the component and do the same to visualize the critical path inside the component.

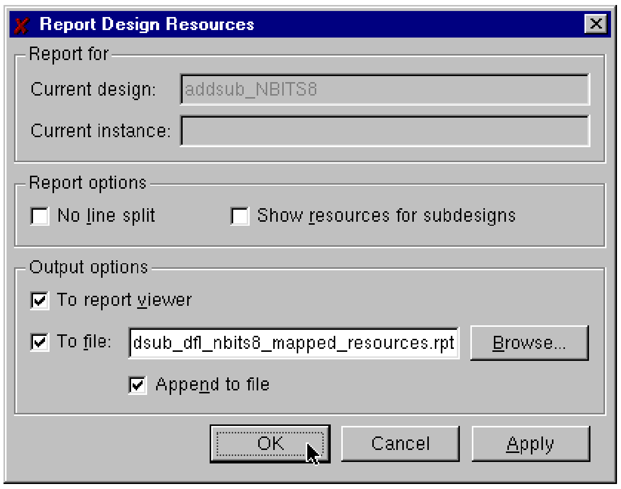


Another useful report is the list of resources used. A resource is an arithmetic or comparison operator read in as part of an HDL design. Resources can be shared during execution of the compile command.

To get a report on the resources used, select the main menu item **Design -> Report Design Resources...**.

Save the report in the file
SYN/RPT/addsub_dfl_nbits8_mapped_resources.rpt
as well as in the report viewer.

Click **OK**.



Report Design Resources

Report for
Current design: addsub_NBITS8
Current instance:

Report options
☐ No line split ☐ Show resources for subdesigns

Output options
☑ To report viewer
☑ To file: dsub_dfl_nbits8_mapped_resources.rpt
☑ Append to file

A new window and the console now display the report:

```
**********************************
Report : resources
Design : addsub_NBITS8
Version: X-2005.09-SP4
Date   : Thu Nov 30 11:20:15 2006
**********************************

Resource Sharing Report for design addsub_NBITS8 in file
        /home/vachoux/educ/edabd/ADDSUB_06/HDL/RTL/addsub_dfl.vhd
=============================================================================
|          |              |            | Contained      |                   |
| Resource | Module       | Parameters | Resources      | Contained Operations |
=============================================================================
| r248     | DW01_addsub  | width=8    |                | add_30 sub_30     |
=============================================================================

Implementation Report
=============================================================================
|                   |                  | Current            | Set            |
| Cell              | Module           | Implementation     | Implementation |
=============================================================================
| r248              | DW01_addsub      | rpl                |                |
=============================================================================
No multiplexors to report
```

You can see that the inferred arithmetic component is implemented as a ripple-carry (rpl) architecture. The tool uses the so-called DesignWare library which contains a number of predefined HDL models of blocks (arithmetic, etc.) with several possible architectures for each block. The best architecture is selected to meet the design constraints.

3.8 VHDL/Verilog gate-level netlist generation and post-synthesis timing data (SDF) extraction

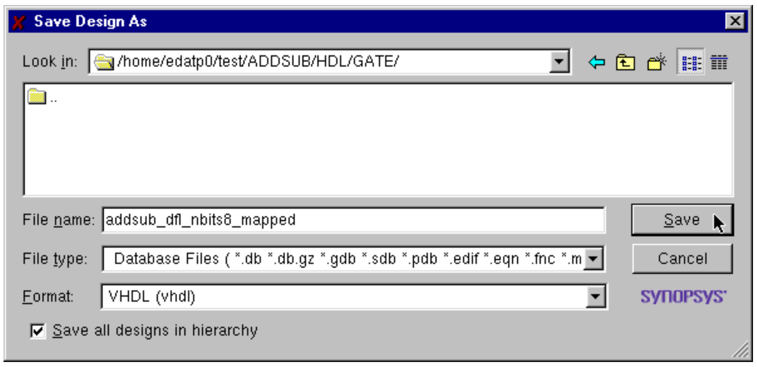
This step generates a VHDL model of the mapped design for simulation and a Verilog model of the same design to be used as input to the placement and routing tool. It also generates a SDF (Standard Delay Format) file that includes the gate delays. Care should be taken to use the right naming scheme when generating the SDF file, otherwise the back-annotation of the delays onto the VHDL or Verilog netlists for simulation will fail. Here we only consider the back-annotation of VHDL netlists.

Before generating the VHDL netlist, it is required to apply some VHDL naming rules to the design. This is done by entering the following command in the console (be sure that the entity addsub_NBITS8 is selected in the hierarchy window):

```
change_names -hierarchy -rules vhdl -verbose
```

Save the mapped design in the file addsub_dfl_nbits8_mapped.vhdl in the directory HDL/GATE.

Note: the dialog window creates VHDL files with the .vhdl extension rather than .vhd as used so far.


Save Design As

Look in: /home/edatp0/test/ADDSUB/HDL/GATE/

File name: addsub_dfl_nbits8_mapped

File type: Database Files (*.db *.db.gz *.gdb *.sdb *.pdb *.edif *.eqn *.fnc *.m

Format: VHDL (vhdl)

☑ Save all designs in hierarchy

Click **Save**. The console now echoes the equivalent command line:

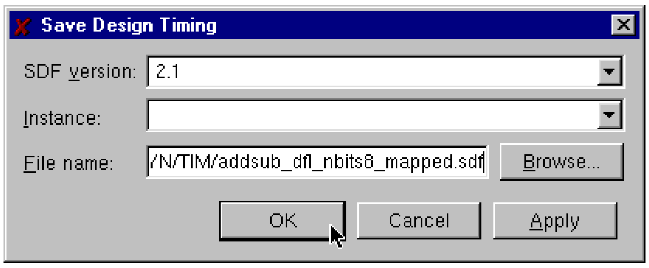
```
write -hierarchy -format vhdl
      -output .../ADDSUB/HDL/GATE/addsub_dfl_nbits8_mapped.vhdl
```

The console issues many warnings about the creation of dummy nets. This is because QN ports of inferred flip-flops are not used and the VHDL generator does not want to have open ports in the generated file.

To generate the SDF file, select the main menu item **File -> Save info -> Design Timing...**.

Save the timing file in the file
SYN/TIMSDF/addsub_dfl_nbits8_mapped.sdf

Click **OK**. The console now includes the equivalent command line and some informational message:


Save Design Timing

SDF version: 2.1

Instance:

File name: /N/TIM/addsub_dfl_nbits8_mapped.sdf

```
write_sdf -version 2.1 SYN/TIM/addsub_dfl_nbits8_mapped.sdf
Information: Annotated 'cell' delays are assumed to include load delay.
```

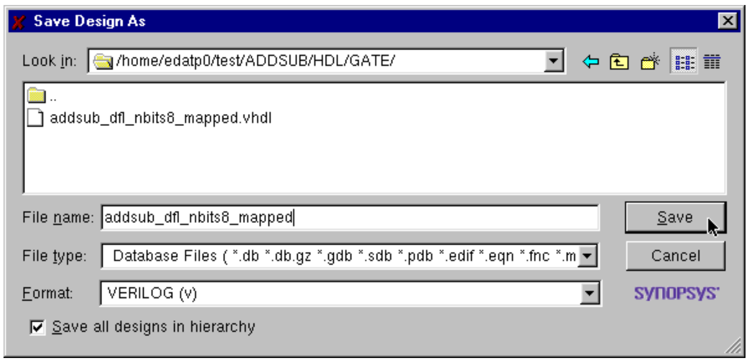
The informational message says that the estimated interconnect delays are actually included in the SDF file as part of the cell delays. The generated SDF file actually includes a list of interconnect delays of zero values.

Before generating the Verilog netlist, it is better to reload the database and apply specific Verilog naming rules to the design. This is done by selecting **File -> Remove All Designs** from the main menu, then reading the database file ./SYN/DB/addsub_dfl_nbits8_mapped.db, and entering the following command in the console (be sure that the entity addsub_NBITS8 is selected in the hierarchy window):

```
change_names -hierarchy -rules verilog -verbose
```

Save the mapped design in the file addsub_dfl_nbits8_mapped.v in the directory HDL/GATE.

Click **Save**. The console now echoes the equivalent command line:


Save Design As

Look in: /home/edatp0/test/ADDSUB/HDL/GATE/

..
addsub_dfl_nbits8_mapped.vhdl

File name: addsub_dfl_nbits8_mapped

File type: Database Files (*.db *.db.gz *.gdb *.sdb *.pdb *.edif *.eqn *.fnc *.m

Format: VERILOG (v)

☑ Save all designs in hierarchy

```
write -hierarchy -format verilog
      -output .../ADDSUB/HDL/GATE/addsub_dfl_nbits8_mapped.v
```

3.9 Design constraints generation for placement and routing

Both design environment and design constraint definitions may be stored in a format that can be read by other Synopsys tools such as PrimeTime or other EDA tool such as Cadence Silicon EnsembleEncounter. The following command creates a new file that includes the design constraints that have been defined for synthesis in Tcl format:

```
write_sdc -nosplit SYN/SDC/addsub_dfl_nbits8_mapped.sdc
```

It is important to do that step after the Verilog naming rules have been applied to the mapped design (see 3.8), otherwise there could be discrepencies on port/signal names between the netlist and the constraint file.

3.10 Design optimization with tighter constraints

It is possible to let the synthesizer infer another faster adder architecture, e.g., a carry look-ahead architecture, by shortening the clock period. The goal here is to redo some steps in this chapter and to compare the results with the ones obtained with the initially “slow” clock.

1. Read the elaborated design. It is not necessary to re-analyze the VHDL sources.
2. Specifiy the clock with a 5 ns period.
3. Save the new elaborated entity in the file SYN/DB/addsub_dfl_nbits8_5ns_elab.db.
4. Map and optimize the design.
5. Save the mapped design in the file SYN/DB/addsub_dfl_nbits8_5ns_mapped.db.
6. Get the new area, timing and resources reports. Compare with the reports you got for the 10 ns clock period.
7. Generate the VHDL gate-level netlist in HDL/GATE/addsub_dfl_nbits8_5ns_mapped.vhdl and the associated SDF timing data file in SYN/SDFTIM/addsub_dfl_nbits8_5ns_mapped.sdf.
8. Do a post-synthesis simulation.
9. Generate the Verilog gate-level netlist in HDL/GATE/dfl_nbits8_5ns_mapped.v.
10. Save the design constraints for placement and routing in the file SYN/SDC/addsub_dfl_nbits8_5ns_mapped.sdc

3.11 Using scripts

It is much more convenient to use scripts and to run the synthesis tool in batch mode when the design complexity increases. Scripts also conveniently capture the synthesis flow and make it reusable. Synopsys Design Compiler supports the Tcl language for building scripts.

An example of such a script for the synthesis of the adder-subtrator design has been installed in the SYN/BIN directory (see “1.5 VHDL example: Adder-subtractor”). The script must be run from the project top directory and it assumed a directory organization as described in “1.2 Design project organisation”.

To run the Tcl script, execute the following command in a Unix shell:

```
[80]...-ADDSUB> dc_shell -f SYN/BIN/addsub_syn.tcl
```

When the script finishes executing, the dc_shell environment is still active so you can enter other dc_shell commands. Enter quit or exit to return to the Unix shell.

The script is given below. It may be modified to define design information and constraints and to control the flow to some extent.

```
#
#   Synopsys DC shell script for adder-subtractor.
#
#   Process: AMS 0.35u CMOS (C35), Hit-Kit 3.70
#

#----------------------------------------------------------------------------
#   It is assumed that a project directory structure has already been
#   created using 'create_project' and that this synthesis script is
#   executed from the project root directory $PROJECT_DIR
#----------------------------------------------------------------------------
set PROJECT_DIR [pwd]

#----------------------------------------------------------------------------
#   Design related information (can be changed)
#----------------------------------------------------------------------------
set VHDL_ENTITY addsub
set VHDL_ARCH dfl
set NBITS 8
set CLK_NAME clk
# all time values are in ns
set CLK_PERIOD 10;
set INPUT_DELAY 2;
set OUTPUT_DELAY 2;
set OPERATING_COND WORST-IND

#----------------------------------------------------------------------------
#   Flags that drive the script behavior (can be changed)
#
#   DB_FORMAT (db | ddc)
#      if db, use the old DB format to store design information
#      if ddc, use the new XG format to store design information (recommended)
#   SHARE_RESOURCES (0 | 1)
#      if 1, force the tool to share resources as much as possible
#      if 0, no resource sharing
#   COMPILE_SIMPLE (0 | 1)
#      if 1, only do a single compile with default arguments
#      if 0, do a two-step compilation with ungrouping in between
#   OPT (string)
#      can be used to have different mapped file names
#----------------------------------------------------------------------------
set DB_MODE ddc
set SHARE_RESOURCES 1
set COMPILE_SIMPLE 1
set OPT "_clk10ns"        ;# to denote the 10ns clock period case

#----------------------------------------------------------------------------
#   File names
#----------------------------------------------------------------------------
set SOURCE_FILE_NAME          ${VHDL_ENTITY}_${VHDL_ARCH}
set ROOT_FILE_NAME            ${VHDL_ENTITY}_${VHDL_ARCH}_nbits${NBITS}
set VHDL_SOURCE_FILE_NAME     ${SOURCE_FILE_NAME}.vhd
set ELAB_FILE_NAME            ${ROOT_FILE_NAME}_elab
set MAPPED_FILE_NAME          ${ROOT_FILE_NAME}${OPT}_mapped
set DB_ELAB_FILE_NAME         ${ELAB_FILE_NAME}.$DB_MODE
set DB_MAPPED_FILE_NAME       ${MAPPED_FILE_NAME}.$DB_MODE
set VHDL_NETLIST_FILE_NAME    ${MAPPED_FILE_NAME}.vhd
set VLOG_NETLIST_FILE_NAME    ${MAPPED_FILE_NAME}.v
set SDF_FILE_NAME             ${MAPPED_FILE_NAME}.sdf
set SDC_FILE_NAME             ${MAPPED_FILE_NAME}.sdc
set RPT_AREA_FILE_NAME        ${MAPPED_FILE_NAME}_area.rpt
set RPT_TIMING_FILE_NAME      ${MAPPED_FILE_NAME}_timing.rpt
set RPT_RESOURCES_FILE_NAME   ${MAPPED_FILE_NAME}_resources.rpt
set RPT_REFERENCES_FILE_NAME  ${MAPPED_FILE_NAME}_references.rpt
set RPT_CELLS_FILE_NAME       ${MAPPED_FILE_NAME}_cells.rpt

#----------------------------------------------------------------------------
#   Absolute paths
```

```
#------------------------------------------------------------------------------
set VHDL_SOURCE_FILE       ${PROJECT_DIR}/HDL/RTL/${VHDL_SOURCE_FILE_NAME}
set VHDL_NETLIST_FILE      ${PROJECT_DIR}/HDL/GATE/${VHDL_NETLIST_FILE_NAME}
set VLOG_NETLIST_FILE      ${PROJECT_DIR}/HDL/GATE/${VLOG_NETLIST_FILE_NAME}
set DB_ELAB_FILE           ${PROJECT_DIR}/SYN/DB/${DB_ELAB_FILE_NAME}
set DB_MAPPED_FILE         ${PROJECT_DIR}/SYN/DB/${DB_MAPPED_FILE_NAME}
set SDF_FILE               ${PROJECT_DIR}/SYN/TIM/${SDF_FILE_NAME}
set SDC_FILE               ${PROJECT_DIR}/SYN/SDC/${SDC_FILE_NAME}
set RPT_AREA_FILE          ${PROJECT_DIR}/SYN/RPT/${RPT_AREA_FILE_NAME}
set RPT_TIMING_FILE        ${PROJECT_DIR}/SYN/RPT/${RPT_TIMING_FILE_NAME}
set RPT_RESOURCES_FILE     ${PROJECT_DIR}/SYN/RPT/${RPT_RESOURCES_FILE_NAME}
set RPT_REFERENCES_FILE    ${PROJECT_DIR}/SYN/RPT/${RPT_REFERENCES_FILE_NAME}
set RPT_CELLS_FILE         ${PROJECT_DIR}/SYN/RPT/${RPT_CELLS_FILE_NAME}

#------------------------------------------------------------------------------
#   Analyze RTL source
#------------------------------------------------------------------------------
analyze -format vhdl -lib WORK $VHDL_SOURCE_FILE

#------------------------------------------------------------------------------
#   Elaborate design
#------------------------------------------------------------------------------
elaborate $VHDL_ENTITY \
   -arch $VHDL_ARCH \
   -lib DEFAULT -update \
   -param [set NBITS $NBITS]

#------------------------------------------------------------------------------
#   Define environment
#------------------------------------------------------------------------------
set_operating_conditions -library c35_CORELIB $OPERATING_COND

#------------------------------------------------------------------------------
#   Define constraints
#------------------------------------------------------------------------------
create_clock -name $CLK_NAME -period $CLK_PERIOD [get_ports $CLK_NAME]

set_input_delay $INPUT_DELAY -clock $CLK_NAME [list [all_inputs]]
set_output_delay $OUTPUT_DELAY -clock $CLK_NAME [list [all_outputs]]

set_max_area 0

# Use only plain DFF cells
set_dont_use [list c35_CORELIB.db:c35_CORELIB/DFE* \
                   c35_CORELIB.db:c35_CORELIB/DFS* \
                   c35_CORELIB.db:c35_CORELIB/TF* \
                   c35_CORELIB.db:c35_CORELIB/JK*]

set_fix_multiple_port_nets -all

#------------------------------------------------------------------------------
#   Set resource allocation and implementation
#------------------------------------------------------------------------------
set_resource_implementation use_fastest
if { $SHARE_RESOURCES } {
   set_resource_allocation area_only
} else {
   set_resource_allocation none
}

#------------------------------------------------------------------------------
#   Save elaborated design and constraints
#------------------------------------------------------------------------------
write -hierarchy -format $DB_MODE -output $DB_ELAB_FILE
```

```
#------------------------------------------------------------------------------
#   Map design to gates
#------------------------------------------------------------------------------
if { $COMPILE_SIMPLE } {
   compile
} else {
   compile -map_effort medium -area_effort medium
   ungroup -all -flatten
   compile -incremental -map_effort high
}

#------------------------------------------------------------------------------
#   Save mapped design
#------------------------------------------------------------------------------
write -hierarchy -format $DB_MODE -output $DB_MAPPED_FILE

#------------------------------------------------------------------------------
#   Generate reports
#------------------------------------------------------------------------------
report_area -nosplit > $RPT_AREA_FILE
report_timing -path full \
              -delay max \
              -nworst 1 \
              -max_paths 1 \
              -significant_digits 2 \
              -nosplit \
              -sort_by group \
              > $RPT_TIMING_FILE
report_resources -nosplit -hierarchy > $RPT_RESOURCES_FILE
report_reference -nosplit > $RPT_REFERENCES_FILE
report_cell -nosplit > $RPT_CELLS_FILE

#------------------------------------------------------------------------------
#   Generate VHDL netlist
#------------------------------------------------------------------------------
change_names -rule vhdl -hierarchy -verbose
write -format vhdl -hierarchy -output $VHDL_NETLIST_FILE

#------------------------------------------------------------------------------
#   Generate SDF data
#------------------------------------------------------------------------------
write_sdf -version 2.1 $SDF_FILE

#------------------------------------------------------------------------------
#   Generate Verilog netlist
#
#   The design is reloaded from scratch to avoid potential naming problems
#   when using the netlist for placement and routing
#------------------------------------------------------------------------------
remove_design -all
read_file -format $DB_MODE $DB_MAPPED_FILE
change_names -rule verilog -hierarchy -verbose
write -format verilog -hierarchy -output $VLOG_NETLIST_FILE

#------------------------------------------------------------------------------
#   Save system constraints
#------------------------------------------------------------------------------
write_sdc -nosplit $SDC_FILE
```

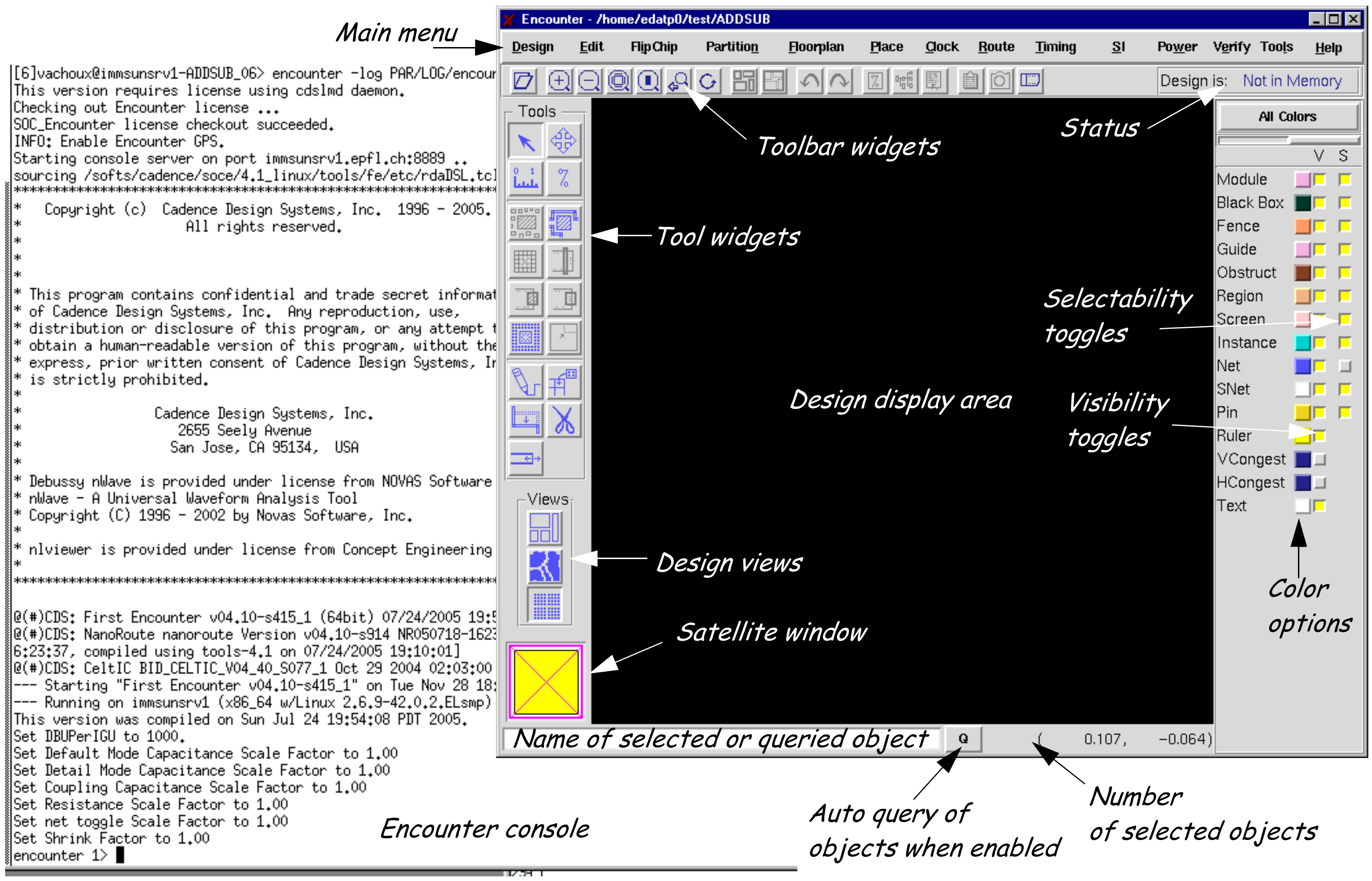
Chapter 4: Standard cell placement and routing

This chapter presents the main steps to perform the placement and the routing of the synthesized gate-level netlist using standard cells from the AMS design kit. The tool used here is Cadence Encounter. The **cdsdoc** on-line command gives access to the Cadence documentation. The tool also has a Help menu.

4.1 Starting the Encounter graphical environment

To start the Encounter environment, enter the **encounter** command in a new Unix shell:

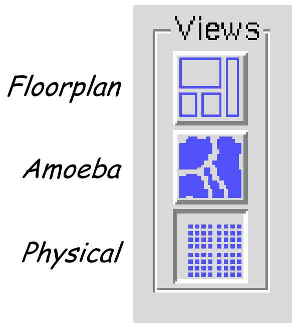
```
[85]vachoux@lsmsun1-ADDSUB> encounter -log PAR/LOG/encounter -overwrite
```



```
[6]vachoux@immsunsrv1-ADDSUB_06> encounter -log PAR/LOG/encour
This version requires license using cdslmd daemon.
Checking out Encounter license ...
SOC_Encounter license checkout succeeded.
INFO: Enable Encounter GPS.
Starting console server on port immsunsrv1.epfl.ch:8889 ..
sourcing /softs/cadence/soce/4.1_linux/tools/fe/etc/rdaDSL.tcl
*****************************************************************
*   Copyright (c)  Cadence Design Systems, Inc.  1996 - 2005.
*                        All rights reserved.
*
*
*
* This program contains confidential and trade secret informat
* of Cadence Design Systems, Inc.  Any reproduction, use,
* distribution or disclosure of this program, or any attempt t
* obtain a human-readable version of this program, without the
* express, prior written consent of Cadence Design Systems, Ir
* is strictly prohibited.
*
*                   Cadence Design Systems, Inc.
*                      2655 Seely Avenue
*                     San Jose, CA 95134,  USA
*
* Debussy nWave is provided under license from NOVAS Software
* nWave - A Universal Waveform Analysis Tool
* Copyright (C) 1996 - 2002 by Novas Software, Inc.
*
* nlviewer is provided under license from Concept Engineering
*
*****************************************************************


@(#)CDS: First Encounter v04.10-s415_1 (64bit) 07/24/2005 19:5
@(#)CDS: NanoRoute nanoroute Version v04.10-s914 NR050718-1623
6:23:37, compiled using tools-4.1 on 07/24/2005 19:10:01]
@(#)CDS: CeltIC BID_CELTIC_V04_40_S077_1 Oct 29 2004 02:03:00
--- Starting "First Encounter v04.10-s415_1" on Tue Nov 28 18:
--- Running on immsunsrv1 (x86_64 w/Linux 2.6.9-42.0.2.ELsmp)
This version was compiled on Sun Jul 24 19:54:08 PDT 2005.
Set DBUPerIGU to 1000.
Set Default Mode Capacitance Scale Factor to 1.00
Set Detail Mode Capacitance Scale Factor to 1.00
Set Coupling Capacitance Scale Factor to 1.00
Set Resistance Scale Factor to 1.00
Set net toggle Scale Factor to 1.00
Set Shrink Factor to 1.00
encounter 1> █
```

The PAR/LOG directory will contain the log file of the session and a file that includes all commands entered in the session. If the -overwrite switch is not used, both log and command files are incremented at each new session. The Unix shell from which the tool is started is called the Encounter console. The console displays the encounter> prompt. This is where you can enter all Encounter text commands and where the tool displays messages. If you use the console for other actions, e.g., Unix commands, the Encounter session suspends until you finish the action.

The main window includes three different design views that you can toggle during a session: the Floorplan view, the Amoeba view, and the Physical view.



The *Floorplan view* displays the hierarchical module and block guides, connection flight lines, and floorplan objects, including block placement, and power/ground nets. The *Amoeba view* displays the outline of the modules and submodules after placement, showing physical locality of the module. The *Physical view* displays the detailed placements of the module's blocks, standard cells, nets, and interconnects.

The main window includes a satellite window, which identifies the location of the current view in the design display area, relative to the entire design. The chip area is identified by a yellow box, the satellite view is identified by the pink crossbox. When you display an entire chip in the design display area, the satellite crossbox encompasses the chip area yellow box. When you zoom and pan through the chip in the design display area, the satellite crossbox identifies where you are relative to the entire chip.

- To move to an area in the design display area, click and drag on the satellite crossbox.
- To select a new area in the design display area, click and drag on the satellite crossbox.
- To resize an area in the satellite window, click with the Shift key and drag a corner of the crossbox.
- To define a chip area in the satellite window, right-click and drag on an area.

There are a number of *binding keys* available (hit the key when the Encounter GUI is active):

• *b*	display the list of binding keys
• *d*	(de)select or delete objects
• *f*	zoom the display to fit the core area
• *k*	create a ruler
• *K*	remove last ruler displayed
• *q*	display the object attribute editor form for the selected object; click the left-button mouse to select an object, Shift-click to select or deselect an object
• *u*	undo last command
• *U*	redo last command
• *z*	zoom-in 2x
• *Z*	zoom-out 2x
• *Arrows*	pan the display.

Hit CTRL-R to refresh the display.

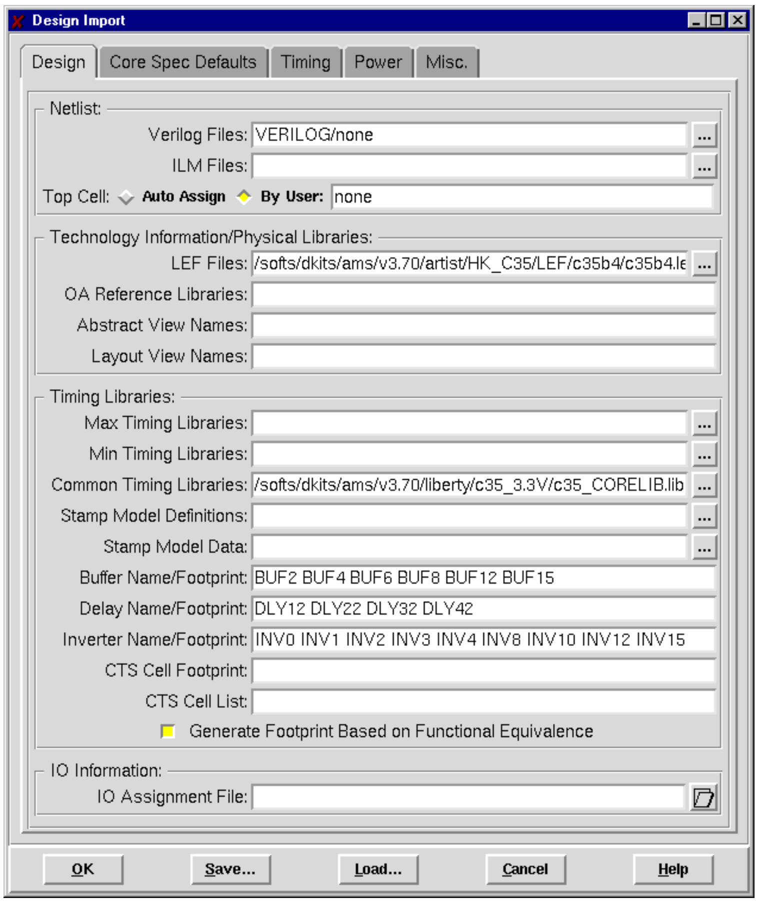
4.2 Design import

Importing the design into Encounter involves specifying the following setup information:

- **Design libraries and files**. This includes information on the technological process and the cell library in the LEF (Layout Exchange Format) format. LEF files provides information such as metal and via layers and via generate rules which is used for routing tasks. They also provide the minimum information on cell layouts for placement and routing.
- **Gate-level netlist**. This relates to the (Verilog) netlist to be placed and routed.
- **Timing libraries**. This includes information on the cell timings (delays, setup/hold times, etc.).
- **Power information**. This relates to the power nets to use in the layout.

To start the design import, select **Design -> Design Import...** in the main menu. Then, click on the **Load...** button and load the file

PAR/CONF/c35b4_std.conf



Netlist:
Verilog Files: VERILOG/none
ILM Files:
Top Cell: Auto Assign [✓ By User:] none

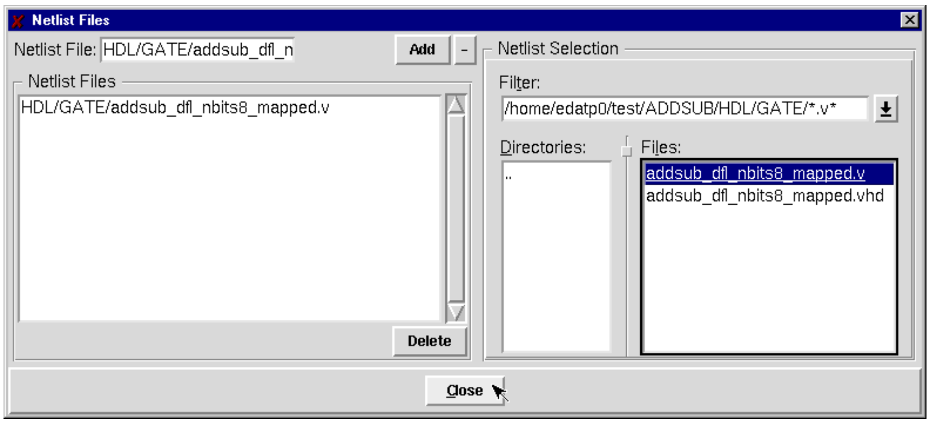
Technology Information/Physical Libraries:
LEF Files: /softs/dkits/ams/v3.70/artist/HK_C35/LEF/c35b4/c35b4.le
OA Reference Libraries:
Abstract View Names:
Layout View Names:

Timing Libraries:
Max Timing Libraries:
Min Timing Libraries:
Common Timing Libraries: /softs/dkits/ams/v3.70/liberty/c35_3.3V/c35_CORELIB.lib
Stamp Model Definitions:
Stamp Model Data:
Buffer Name/Footprint: BUF2 BUF4 BUF6 BUF8 BUF12 BUF15
Delay Name/Footprint: DLY12 DLY22 DLY32 DLY42
Inverter Name/Footprint: INV0 INV1 INV2 INV3 INV4 INV8 INV10 INV12 INV15
CTS Cell Footprint:
CTS Cell List:
Generate Footprint Based on Functional Equivalence

IO Information:
IO Assignment File:

This file defines a basic import configuration. There is a number of additions and changes to bring to the initial configuration. The new configuration will then be saved for future uses.

The first information to add is the netlist. Click on the **...** button on the right of the Verilog Files field. You get a new dialog window with only one pane. Click on the top-right icon to get the full window.



Netlist File: HDL/GATE/addsub_dfl_n
Netlist Files
HDL/GATE/addsub_dfl_nbits8_mapped.v

Netlist Selection
Filter:
/home/edatp0/test/ADDSUB/HDL/GATE/*.v*
Directories:
..
Files:
addsub_dfl_nbits8_mapped.v
addsub_dfl_nbits8_mapped.vhd

Remove the VERILOG/none line in the left pane.

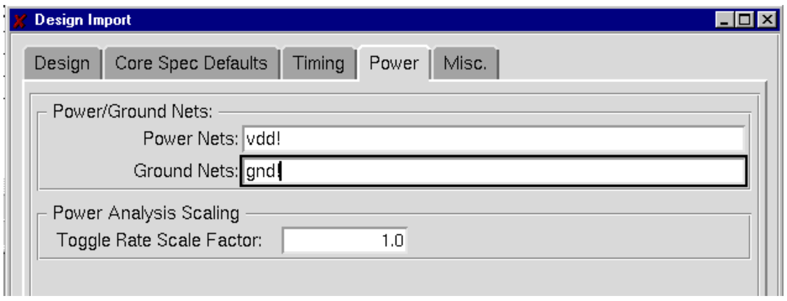
Select the Verilog netlist file HDL/GATE/addsub_dfl_nbits8_mapped.v (or the Verilog netlist you want to place and route), add it to the left pane and close the window. It is assumed here that the imported netlist is the one generated for the 10 ns clock period.

In the Design Import window, select the Auto Assign box to let the tool extract the top cell name from the file. If the Verilog file includes more than one design (more than one top module name), you need to give the name of the top module to use explicitly.

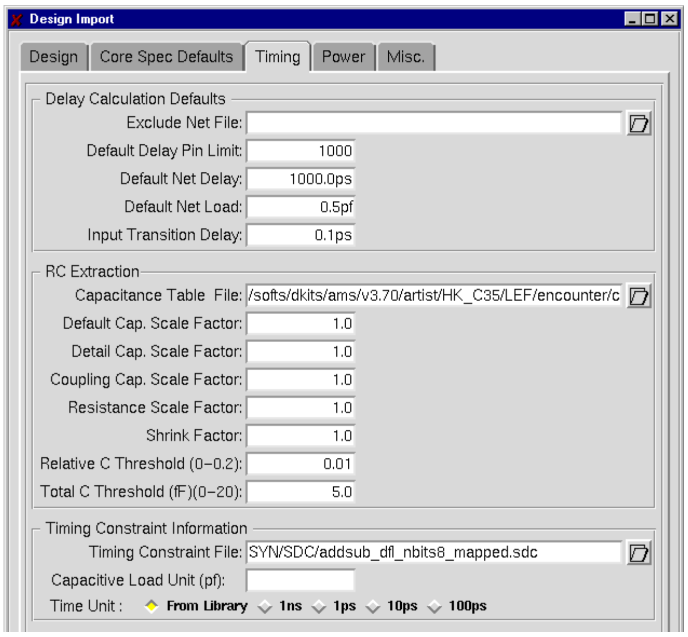
In the "LEF files" and "Common Timing Libraries" fields, you should remove the reference to the IO pad library if you don't intend to include pads, namely:

/softs/dkits/ams/v3.70/artist/HK_C35/LEF/c35b4/IOLIB_4M.lef
/softs/dkits/ams/v3.70/liberty/c35_3.3V/c35_IOLIB_4M.lib

In the Power tag, you can keep only the vdd! and gnd! power nets if you won't add periphery cells. The names of power and ground nets must be the same as the ones used in the LEF file that describes the standard cells.



Power/Ground Nets:
Power Nets: vdd!
Ground Nets: gnd!
Power Analysis Scaling
Toggle Rate Scale Factor: 1.0



Delay Calculation Defaults
Exclude Net File:
Default Delay Pin Limit: 1000
Default Net Delay: 1000.0ps
Default Net Load: 0.5pf
Input Transition Delay: 0.1ps
RC Extraction
Capacitance Table File: /softs/dkits/ams/v3.70/artist/HK_C35/LEF/encounter/c
Default Cap. Scale Factor: 1.0
Detail Cap. Scale Factor: 1.0
Coupling Cap. Scale Factor: 1.0
Resistance Scale Factor: 1.0
Shrink Factor: 1.0
Relative C Threshold (0–0.2): 0.01
Total C Threshold (fF)(0–20): 5.0
Timing Constraint Information
Timing Constraint File: SYN/SDC/addsub_dfl_nbits8_mapped.sdc
Capacitive Load Unit (pf):
Time Unit : [✓ From Library] 1ns 1ps 10ps 100ps

In the Timing tag, you can specify the timing constraints that you used for synthesis.

Select the file that has been generated during logic synthesis (3.6 Design mapping and optimization):

SYN/SDC/addsub_dfl_nbits8_mapped.sdc

Only timing information in the constraint file is actually used by Encounter.

The rest of the settings can be left as is. You can now save the updated configuration in the file PAR/CONF/addsub_nbits8.conf by clicking on the **Save...** button.

Finally, click on **OK**. The configuration is then read in.

To reload a configuration, select **Design -> Design Import...** in the main menu. Then, click on the **Load...** button and load the configuration file from the PAR/CONF directory.

4.3 Floorplan Specification

The floorplan defines the actual form, or aspect ratio, the layout will take, the global and detailed routing grids, the rows to host the core cells and the I/O pad cells (if required), and the location of the corner cells (if required).

Select **Floorplan -> Specify Floorplan...** in the main menu.

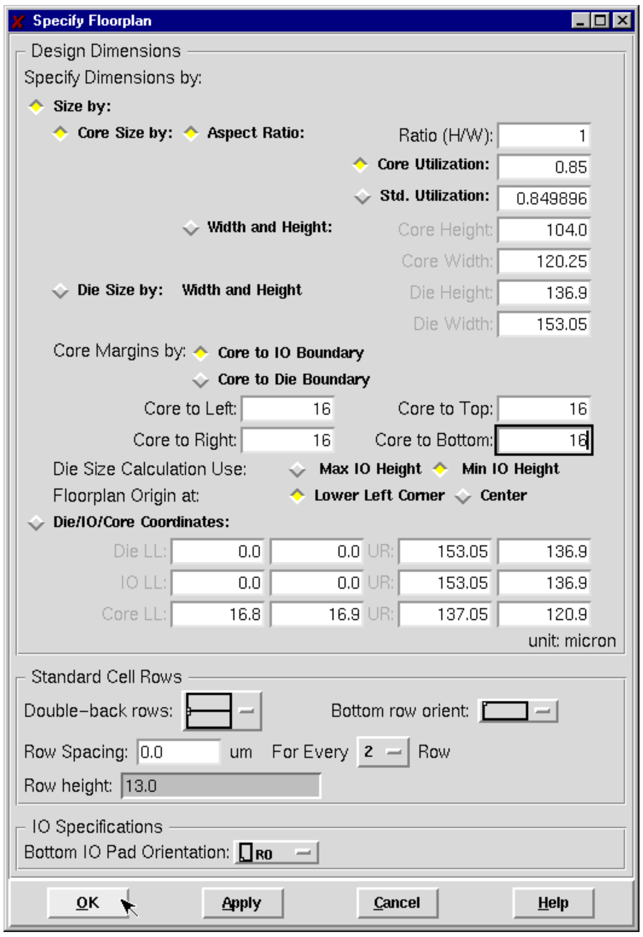
Define an aspect ratio of 1.

The default core utilization is 85%, which means that 15% of the core area will be free for possible future buffer insertions or cell replacements. The other utilization ratio (Std. Utilization) refers to the density of the standard cells.

The space between the I/Os and the core should be large enough to host a number of power and ground rings and possible periphery cells (I/O pads). Define I/O to core distances of 16 microns. This is enough as there will be one power and one ground ring of 4 micron width each and there won't be any pads in the layout.

The cell rows will be abbutted and flipped each two row to share VDD and ground power rails.

Clicking on **Apply** updates the numbers to actual values.



Specify Floorplan

Design Dimensions

Specify Dimensions by:

[✓ Size by:]

[✓ Core Size by:] [✓ Aspect Ratio:] Ratio (H/W): 1

[✓ Core Utilization:] 0.85

Std. Utilization: 0.849896

Width and Height: Core Height: 104.0

Core Width: 120.25

Die Size by: Width and Height Die Height: 136.9

Die Width: 153.05

Core Margins by: [✓ Core to IO Boundary] Core to Die Boundary

Core to Left: 16 Core to Top: 16

Core to Right: 16 Core to Bottom: 16

Die Size Calculation Use: Max IO Height [✓ Min IO Height]

Floorplan Origin at: [✓ Lower Left Corner] Center

Die/IO/Core Coordinates:

Die LL:	0.0	0.0	UR: 153.05	136.9
IO LL:	0.0	0.0	UR: 153.05	136.9
Core LL:	16.8	16.9	UR: 137.05	120.9

unit: micron

Standard Cell Rows

Double-back rows: Bottom row orient:

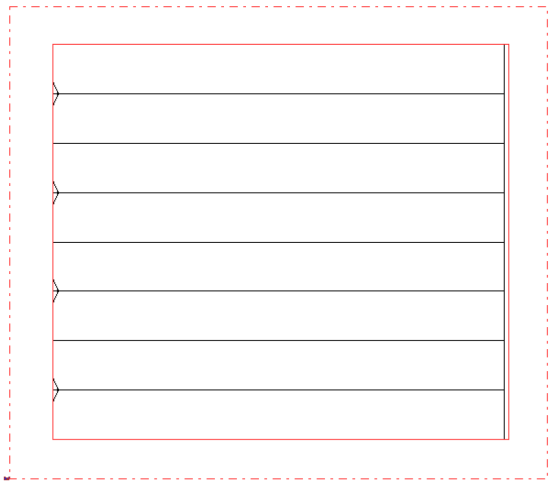
Row Spacing: 0.0 um For Every 2 Row

Row height: 13.0

IO Specifications

Bottom IO Pad Orientation: R0

Click **OK**. The display design area pane now shows the defined floorplan with the required number of rows.



It is a good idea to save the design at that stage to allow restarting here quickly without needing to redo all the previous steps.

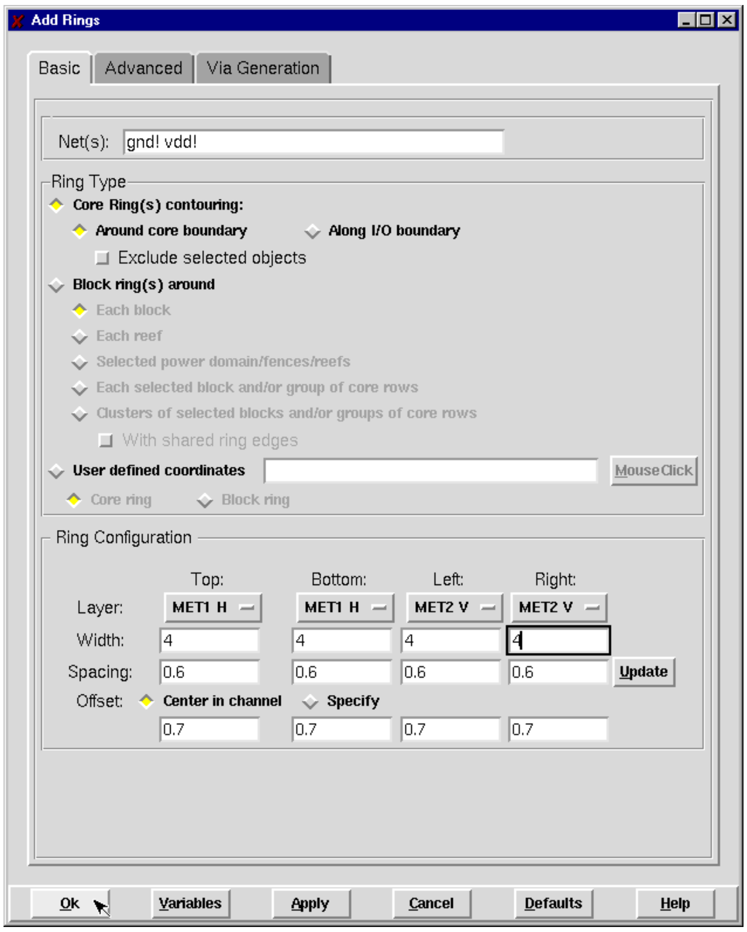
Select **Design -> Save Design...** in the main menu and save the current state in the file
PAR/DB/addsub_nbits8-fplan.enc.
The data are actually saved in the directory
PAR/DB/addsub_nbits8-fplan.enc.dat.

To restore design data, select
Design -> Restore Design... in the main menu and select the .enc file to read in the PAR/DB directory.

4.4 Power ring/stripe creation and routing

This step generates the VDD and ground power rings around the core and optionally adds a number of vertical and/or horizontal power stripes across the core. Stripes ensure a proper power distribution in large cores. They are not strictly required here as the design is small.

Select **Floorplan -> Power Planning -> Add Rings...** in the main menu.



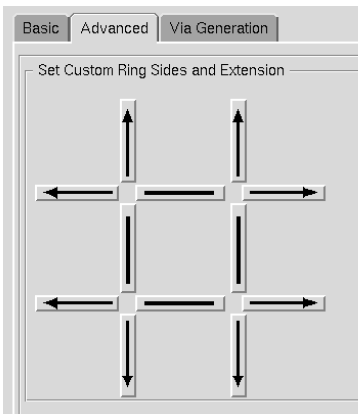
Net(s): gnd! vdd!

Ring Type
- [✓ Core Ring(s) contouring:]
  - [✓ Around core boundary] Along I/O boundary
  - Exclude selected objects
- Block ring(s) around
  - [✓ Each block]
  - Each reef
  - Selected power domain/fences/reefs
  - Each selected block and/or group of core rows
  - Clusters of selected blocks and/or groups of core rows
  - With shared ring edges
- User defined coordinates
  - [✓ Core ring] Block ring

Ring Configuration

	Top:	Bottom:	Left:	Right:
Layer:	MET1 H	MET1 H	MET2 V	MET2 V
Width:	4	4	4	4
Spacing:	0.6	0.6	0.6	0.6
Offset: [✓ Center in channel] Specify	0.7	0.7	0.7	0.7

The Net(s) field defines the number and the kinds of rings from the core. In our case, there will be first a ground ring around the core and a VDD ring around the ground ring. The net names should be consistent with the power net names in the cell LEF file.



The ring configuration defines ring widths of 4 micron spaced by 0.6 micron. The rings will be placed in the center of the channel between the core and the chip boundary (or the IO pads, if any). Check the Center in channel box in the Ring Configuration part.

It is possible to extend the ring segments to reach the core boundary. Click on the Advanced tab and click on the segments you'd like to extend. Other power and ground side trunks can be defined by selecting only horizontal or vertical segments.

Click **OK** to generate the rings.

To add power stripes, select **Floorplan -> Power Planning -> Add Stripes...** in the main menu.

The Net(s) field defines the pattern. Here a single pattern will be generated (select Number of sets and insert 1). Each stripe will be 2 micron wide and the space between them will be 1 micron.

Check Relative from core or selected area and enter the value 60 (micron) in the X from left field. The two stripes will be vertically placed near the center of the core.

It is possible to measure sizes by using the ruler (or hit the *k* binding key). Hit *K* to remove the last ruler or press ESC to remove all rulers.

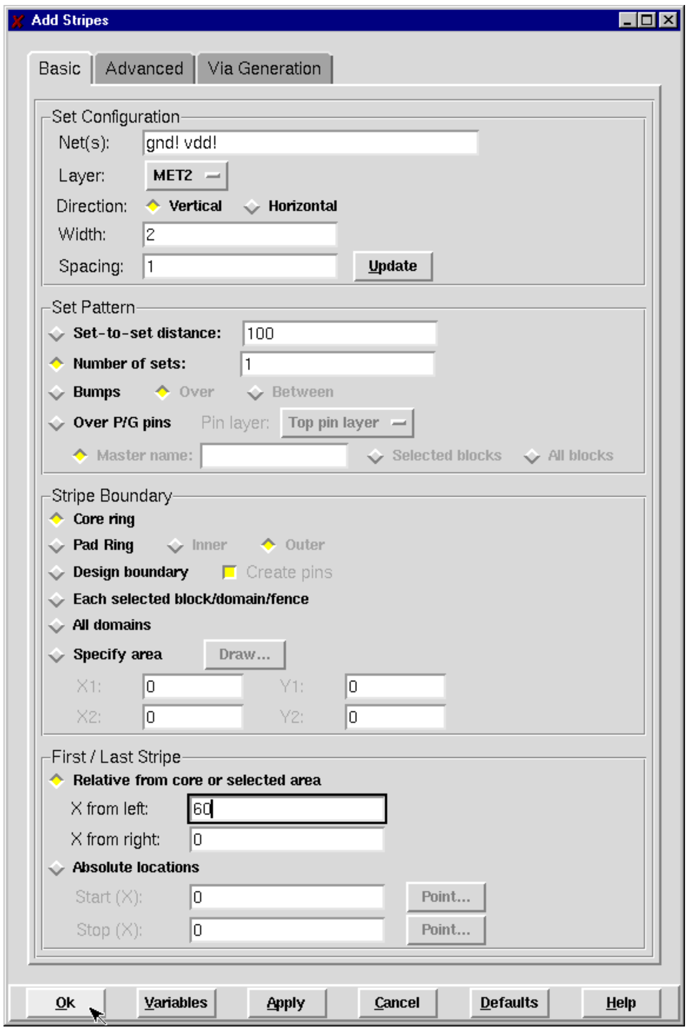
Click **OK** to generate the stripes.

It is possible to remove the stripes by selecting them and hitting the Delete key.

Additionally, the command

Floorplan -> Clear Floorplan...

allows you to delete all or parts of the floorplan objects.



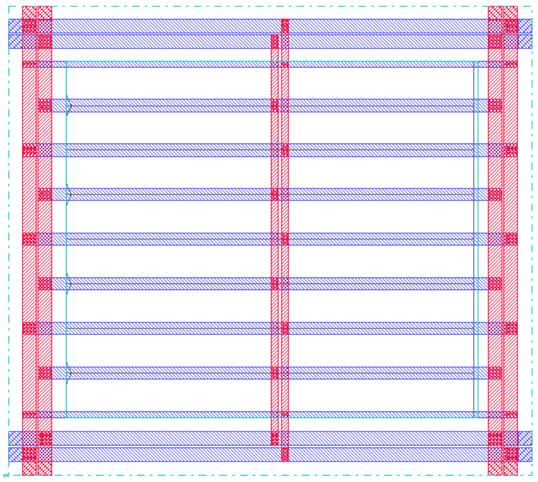
Set Configuration
Net(s): gnd! vdd!
Layer: MET2
Direction: [✓ Vertical] Horizontal
Width: 2
Spacing: 1

Set Pattern
Set-to-set distance: 100
[✓ Number of sets:] 1
Bumps [✓ Over] Between
Over P/G pins Pin layer: Top pin layer
[✓ Master name:] Selected blocks All blocks

Stripe Boundary
[✓ Core ring]
Pad Ring Inner [✓ Outer]
Design boundary [✓ Create pins]
Each selected block/domain/fence
All domains
Specify area Draw...
X1: 0 Y1: 0
X2: 0 Y2: 0

First / Last Stripe
[✓ Relative from core or selected area]
X from left: 60
X from right: 0
Absolute locations
Start (X): 0
Stop (X): 0

Now, it is possible to route the power grid. Select **Route -> SRoute...** in the main menu. All default values are fine. Click **OK** to do the routing. The design now looks like below:

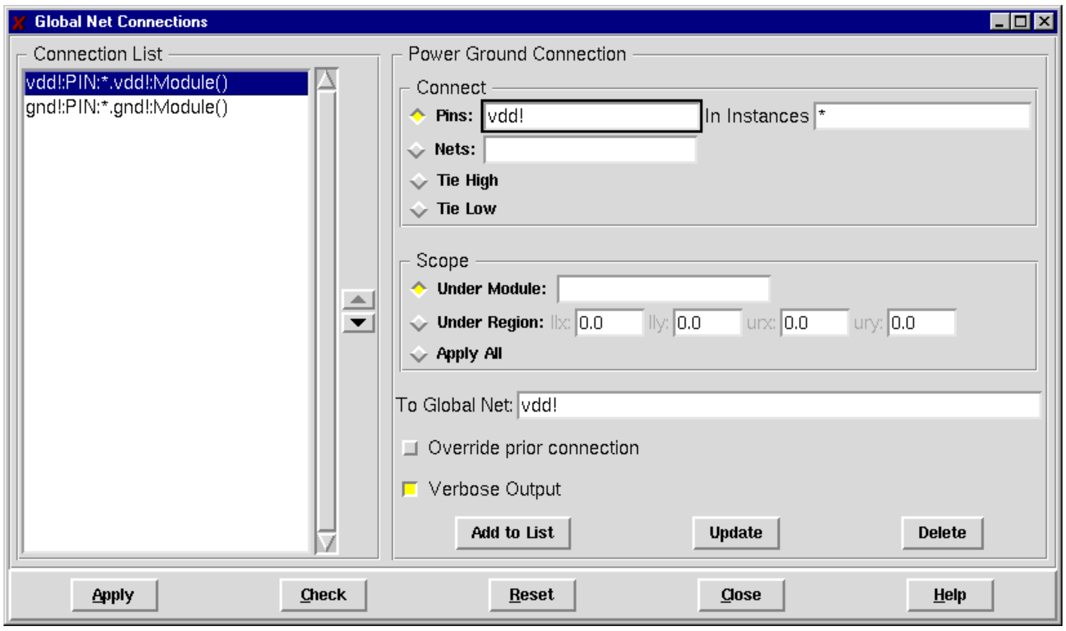


It is recommended to save the new stage of the design. Select **Design -> Save Design...** in the main menu and save the current state in the file PAR/DB/addsub_nbits8-pring.enc.

4.5 Global net connections

This step assigns pins or nets to global power and ground nets. The imported Verilog netlist does not mention any power and ground connections. However, the cells that will be placed do have power/ground pins that will need to be routed to the global power/ground nets defined for the block.

Select **Floorplan -> Global Net Connections...** in the main menu.


Global Net Connections

Connection List:
vdd!:PIN:*.vdd!:Module()
gnd!:PIN:*.gnd!:Module()

Power Ground Connection
Connect: Pins: vdd! In Instances *
Scope: Under Module
Under Region: llx: 0.0 lly: 0.0 urx: 0.0 ury: 0.0
To Global Net: vdd!
Override prior connection
Verbose Output

The left pane (Connection List) is initially empty. For each VDD and ground net:
- Check the Pins field and enter the pin name (vdd! or gnd!).
- Fill the To Global Net field with either vdd! or gnd!.
- Click on Add to List. The left pane now includes the related global net connection.

Click on **Apply** and then on **Close**.

The Encounter console then reports on the number of power pin to global net connections that have been made (the actual number may vary depending on the number of rows defined in the floorplan):

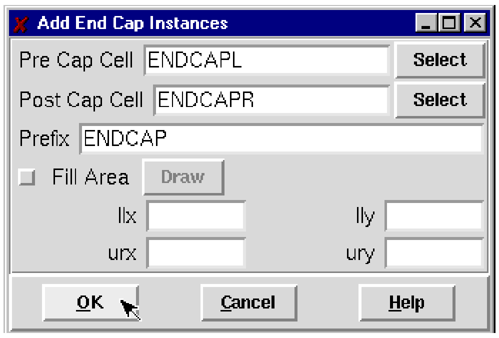
```
44 new pwr-pin connections were made to global net 'vdd!'.
44 new gnd-pin connections were made to global net 'gnd!'.
```

4.6 CAP cell placement

The AMS design kit requires that physical-only termination cells must be placed at each row ends. The so-called CAP cells are used to properly bias the P+ and N+ substrates.

To do this, select **Place -> Filler -> Add End Cap...** in the main menu. Then, select the ENDACPL cell as the cell to place at the beginning of each row and the ENDCAPR cell as the cell to place at the end of each row.

Click **OK**. The added cells are now visible in the Physical view (you might need to redraw the display).

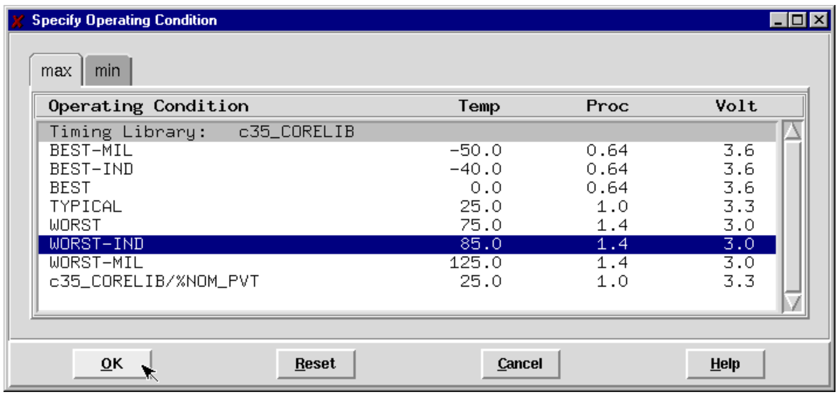

Add End Cap Instances

Pre Cap Cell: ENDCAPL
Post Cap Cell: ENDCAPR
Prefix: ENDCAP
Fill Area
llx lly
urx ury

4.7 Operating conditions definition

The operating conditions define the temperature, process and voltage conditions for the design. They impact the timing calculations and optimizations.

Select the **Timing -> Specify Analysis Conditions -> Specify Operating Condition/PVT...** in the main menu.



Operating Condition	Temp	Proc	Volt
Timing Library: c35_CORELIB			
BEST-MIL	-50.0	0.64	3.6
BEST-IND	-40.0	0.64	3.6
BEST	0.0	0.64	3.6
TYPICAL	25.0	1.0	3.3
WORST	75.0	1.4	3.0
WORST-IND	85.0	1.4	3.0
WORST-MIL	125.0	1.4	3.0
c35_CORELIB/%NOM_PVT	25.0	1.0	3.3

In the max tab, select the WORST-IND operating condition. In the min tab, select the BEST-IND operating condition.

Click **OK**.

The max operating conditions will be used to meet setup timing constraints, while the min operating conditions will be used to meet hold timing constraints.

The Encounter console summarizes the settings:

```
Set Max operating condition to "WORST-IND" defined in Timing Library
"c35_CORELIB"
Process: 1.40 Temperature: 85.000 Voltage: 3.000
Set Min operating condition to "BEST-IND" defined in Timing Library
"c35_CORELIB"
Process: 0.64 Temperature: -40.000 Voltage: 3.600
*** Calculating scaling factor using operating condition:
Name: WORST-IND Process: 1.40 Temperature: 85.000 Voltage: 3.000
```

Running the following command in the Encounter console gives the active operating conditions:

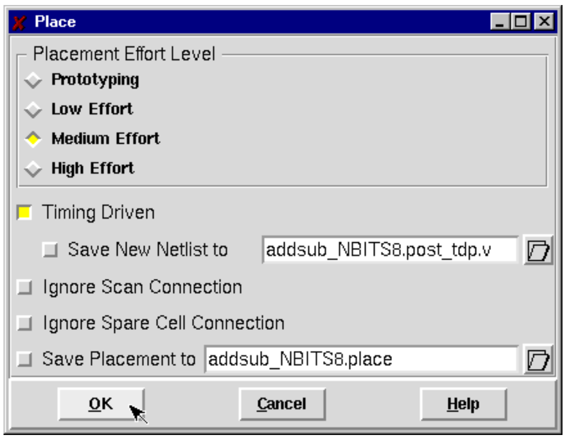
```
encounter 10> getOpCond -v
min: BEST-IND proc: 0.6400 volt: 3.6000 temp: -40.0000
max: WORST-IND proc: 1.4000 volt: 3.0000 temp: 85.0000
```

4.8 Core cell placement

This steps places the cells of the imported Verilog netlist in the rows.

Select **Place -> Place...** in the main menu.

Select the Timing Driven option. This is only valid if timing constraints have been properly defined on the design. This option will optimize the placement of the cells that are on the critical path. Some cell instances may be replaced with cells having lower driving capabilities (downsizing) or stronger driving capabilities (upsizing). Buffers may be also added or deleted. The Encounter console notifies such changes.


Place

Placement Effort Level

- Prototyping
- Low Effort
- [✓ Medium Effort]
- High Effort

[✓ Timing Driven]

Save New Netlist to addsub_NBITS8.post_tdp.v

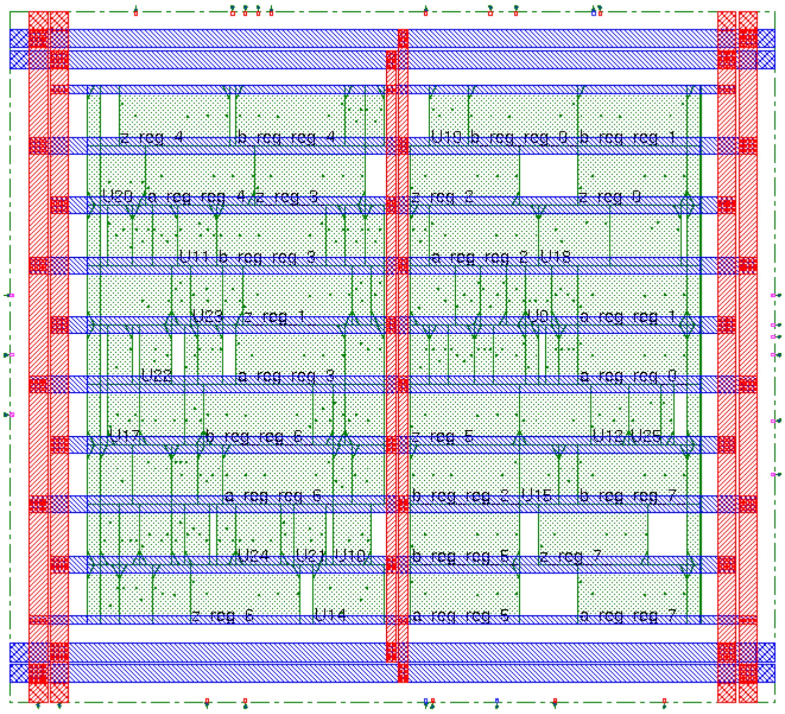
Ignore Scan Connection

Ignore Spare Cell Connection

Save Placement to addsub_NBITS8.place

Click **OK** to do the placement. It may take some time to complete, especially when the placement is timing driven and a high effort level is used

The placement should then look like below:

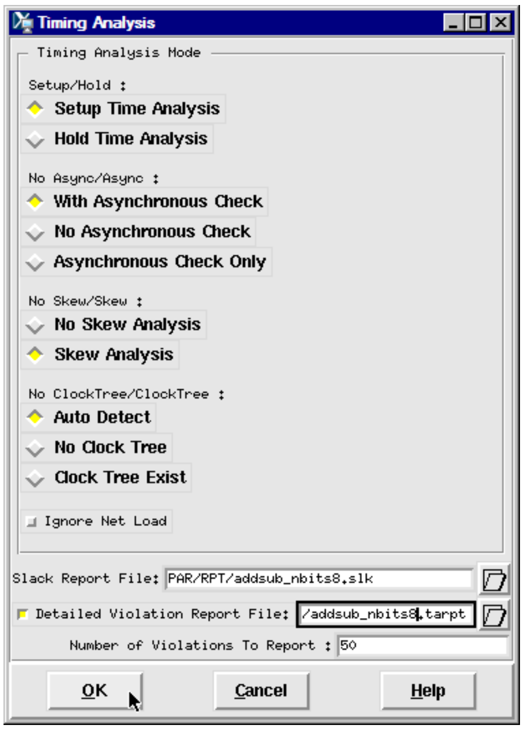


It is recommended to save the new stage of the design. Select **Design -> Save Design...** in the main menu and save the current state in the file PAR/DB/addsub_nbits8-placed.enc.

4.9 Post-placement timing analysis

The timing analysis engine in Encounter can now be run to get a relatively good idea of the timing performances of the design. It actually performs a trial routing and a parasitic extraction based on the current cell placement.

Select **Timing -> Timing Analysis...** in the main menu. Define the path for the slack report file. Click **OK**.


Timing Analysis

Timing Analysis Mode

Setup/Hold :
[✓ Setup Time Analysis]
Hold Time Analysis

No Async/Async :
[✓ With Asynchronous Check]
No Asynchronous Check
Asynchronous Check Only

No Skew/Skew :
No Skew Analysis
[✓ Skew Analysis]

No ClockTree/ClockTree :
[✓ Auto Detect]
No Clock Tree
Clock Tree Exist

Ignore Net Load

Slack Report File: PAR/RPT/addsub_nbits8.slk

[✓ Detailed Violation Report File:] /addsub_nbits8.tarpt

Number of Violations To Report : 50

In the Encounter console window you get a summary of the timing analysis:

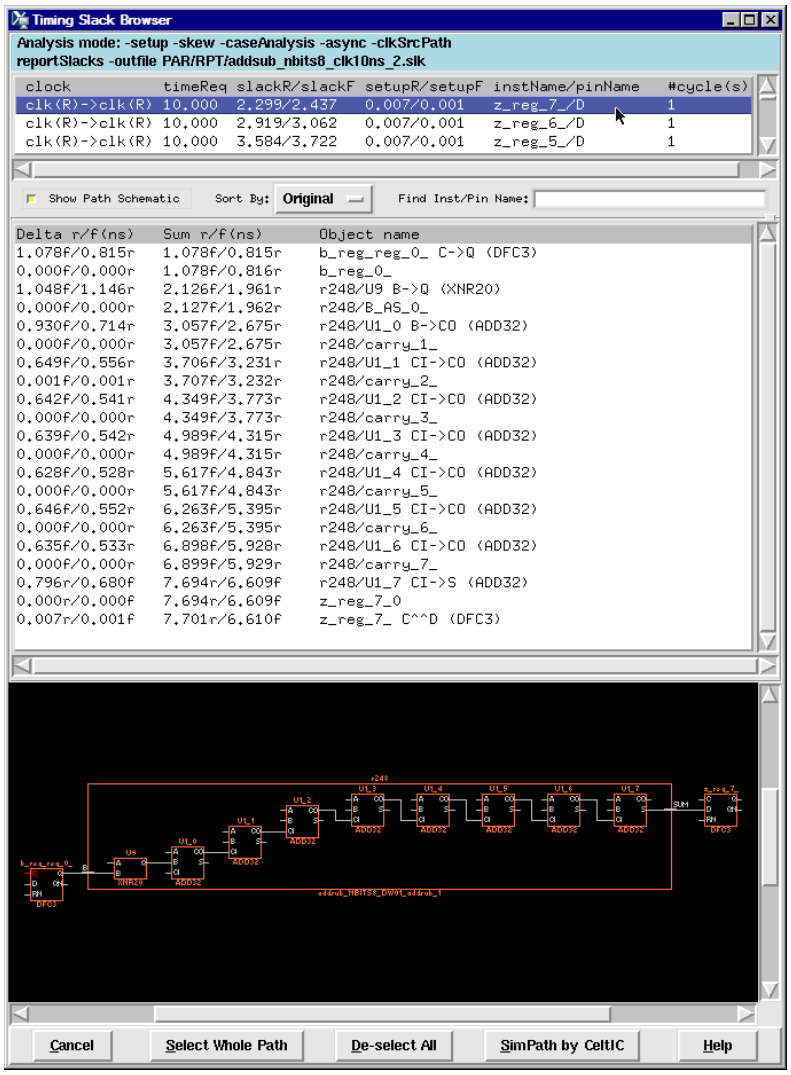
```
*** Dump slack report (0:00:00.0)
**Info: no slack violation path.

Slack File   : PAR/RPT/addsub_nbits8.slk
targetSlack : 0.000 ns
--------------------  -------    -------
 Slack Range (ns)       Count        Sum
--------------------  -------    -------
(  2.299 ~   2.000]         1          1
(  2.000 ~   2.500]         0          1
--------------------  -------    -------
Total negative slacks(TNS)=0
Worst negative slacks(WNS)=2.299
```

The design is not critical as the slack is positive (2.299 ns). Comparing with the timing report obtained during logic synthesis (3.7 Report generation), we notice that the slack here is slightly higher. The difference come from a more accurate computation of parasitics delays.

To get more details on the critical path, select **Timing -> Timing Debug -> Slack Browser...** in the main menu. Load the slack report file generated previously.

Check the Show Path Schematic box to see the path schematic. Double-click on the first path in the upper list.

The critical path is the same as in the one found during logic synthesis.


Timing Slack Browser

Analysis mode: -setup -skew -caseAnalysis -async -clkSrcPath
reportSlacks -outfile PAR/RPT/addsub_nbits8_clk10ns_2.slk

```
clock          timeReq slackR/slackF setupR/setupF instName/pinName  #cycle(s)
clk(R)->clk(R) 10.000  2.299/2.437   0.007/0.001   z_reg_7_/D        1
clk(R)->clk(R) 10.000  2.919/3.062   0.007/0.001   z_reg_6_/D        1
clk(R)->clk(R) 10.000  3.584/3.722   0.007/0.001   z_reg_5_/D        1
```

Show Path Schematic   Sort By: Original   Find Inst/Pin Name:

```
Delta r/f(ns)   Sum r/f(ns)     Object name
1.078f/0.815r   1.078f/0.815r   b_reg_reg_0_ C->Q (DFC3)
0.000f/0.000r   1.078f/0.816r   b_reg_0_
1.048f/1.146r   2.126f/1.961r   r248/U9 B->Q (XNR20)
0.000f/0.000r   2.127f/1.962r   r248/B_AS_0_
0.930f/0.714r   3.057f/2.675r   r248/U1_0 B->CO (ADD32)
0.000f/0.000r   3.057f/2.675r   r248/carry_1_
0.649f/0.556r   3.706f/3.231r   r248/U1_1 CI->CO (ADD32)
0.001f/0.001r   3.707f/3.232r   r248/carry_2_
0.642f/0.541r   4.349f/3.773r   r248/U1_2 CI->CO (ADD32)
0.000f/0.000r   4.349f/3.773r   r248/carry_3_
0.639f/0.542r   4.989f/4.315r   r248/U1_3 CI->CO (ADD32)
0.000f/0.000r   4.989f/4.315r   r248/carry_4_
0.628f/0.528r   5.617f/4.843r   r248/U1_4 CI->CO (ADD32)
0.000f/0.000r   5.617f/4.843r   r248/carry_5_
0.646f/0.552r   6.263f/5.395r   r248/U1_5 CI->CO (ADD32)
0.000f/0.000r   6.263f/5.395r   r248/carry_6_
0.635f/0.533r   6.898f/5.928r   r248/U1_6 CI->CO (ADD32)
0.000f/0.000r   6.899f/5.929r   r248/carry_7_
0.796r/0.680f   7.694r/6.609f   r248/U1_7 CI->S (ADD32)
0.000r/0.000f   7.694r/6.609f   z_reg_7_0
0.007r/0.001f   7.701r/6.610f   z_reg_7_ C^^D (DFC3)
```

Another way to do a timing analysis is to execute the following commands in the Encounter console:

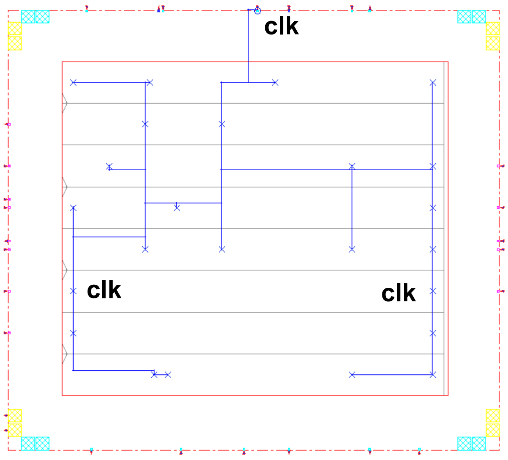
```
encounter 15> setCteReport
encounter 16> reportTA
```

The following report is then displayed in the console:

```
*info: Report constrained paths
*      Path type: max
*      Format: long
*      Operating Condition: WORST-IND
*      Process: 1.4000
*      Voltage: 3.0000
*      Temperature: 85.0000
*      Time Unit: 1ns
*      Capacitance Unit: 1.000000pf
-----------------------------------------------------------------------------
Path #: 1
Startpoint: b_reg_reg_0_/Q
            (clocked by clk R)
Endpoint: z_reg_7_/D
            (Setup time: 0.007, clocked by clk R)
Data required time: 9.993
Data arrival Time: 7.705
Slack: 2.288

Object name             Delta r/f (ns)   Sum r/f (ns)     Slew (ns)   Load (pf)
-----------------------------------------------------------------------------
clk                       0.000r/-f      0.000r/-f      0.000r/0.000f   0.174
b_reg_reg_0_ C->Q (DFC3)  1.078f/-r      1.078f/-r      0.000f/0.000r   0.016
r248/U9 B->Q (XNR20       1.059f/-r      2.137f/-r      0.267f/0.318r   0.041
r248/U1_0 B->CO (ADD32)   0.930f/-r      3.068f/-r      1.179f/2.156r   0.033
r248/U1_1 CI->CO (ADD32)  0.649f/-r      3.717f/-r      0.417f/0.549r   0.039
r248/U1_2 CI->CO (ADD32)  0.643f/-r      4.360f/-r      0.455f/0.607r   0.035
r248/U1_3 CI->CO (ADD32)  0.639f/-r      4.999f/-r      0.430f/0.569r   0.036
r248/U1_4 CI->CO (ADD32)  0.628f/-r      5.628f/-r      0.438f/0.584r   0.033
r248/U1_5 CI->CO (ADD32)  0.646f/-r      6.274f/-r      0.417f/0.551r   0.038
r248/U1_6 CI->CO (ADD32)  0.635f/-r      6.909f/-r      0.455f/0.612r   0.033
r248/U1_7 CI->S (ADD32)   0.796r/-f      7.705r/-f      0.560r/0.423f   0.010
z_reg_7_ D (DFC3)         0.000r/-f      7.705r/-f      0.279r/0.240f   0.010
```

4.10 Clock tree synthesis (optional)

As the paths that will propagate the clock signal in the design are not necessarily balanced, some registers may receive the active clock edge later than others (clock skew) and may therefore violate the assumed synchronous design operation. For example, the original clock tree we can get from the previously placed design is shown below.

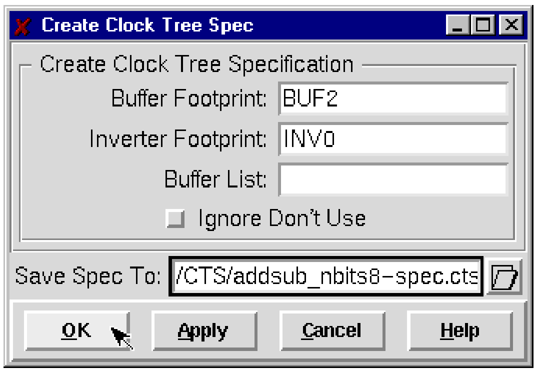


To create a balanced clock tree, you have first to create a clock tree specification file. Encounter can create a first draft version of the file you can then edit to include design specific data.

Select **Clock -> Create Clock Tree Spec...** in the main menu.

Select the BUF2 and INV0 cells for footprints and save the specification in the file
PAR/CTS/addsub_nbits8-spec.cts.

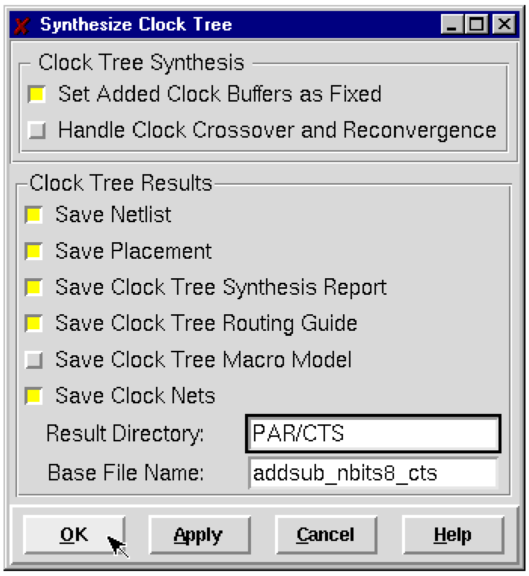
Click **OK** to generate the specification file. You can then edit the file to change timing values such as the maximum allowed clock skew (300ps by default).



Buffer Footprint: BUF2
Inverter Footprint: INV0
Buffer List:
☐ Ignore Don't Use
Save Spec To: /CTS/addsub_nbits8-spec.cts

The next step is to load the clock tree specification file. Select **Clock -> Specify Clock Tree ...** in the main menu and select the file PAR/CTS/addsub_nbits8-spec.cts that has been previously created (and possibly edited).

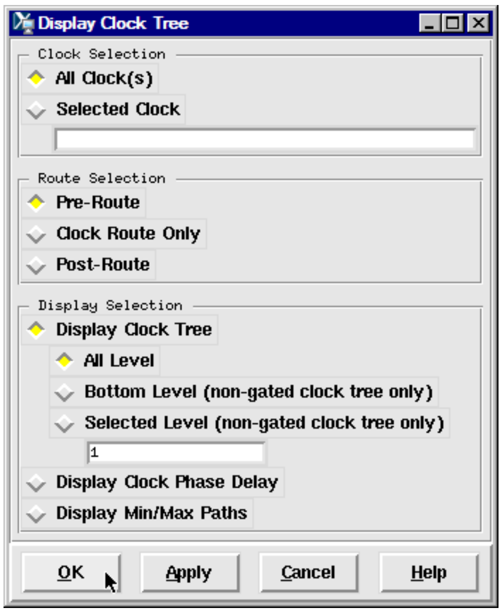
To create the clock tree, select
Clock -> Synthesize Clock Tree ... in the main menu.
Define the result directory as PAR/CTS and the base file name as addsub_nbits8_cts.
Click **OK** to create the clock tree.



☑ Set Added Clock Buffers as Fixed
☐ Handle Clock Crossover and Reconvergence

☑ Save Netlist
☑ Save Placement
☑ Save Clock Tree Synthesis Report
☑ Save Clock Tree Routing Guide
☐ Save Clock Tree Macro Model
☑ Save Clock Nets
Result Directory: PAR/CTS
Base File Name: addsub_nbits8_cts

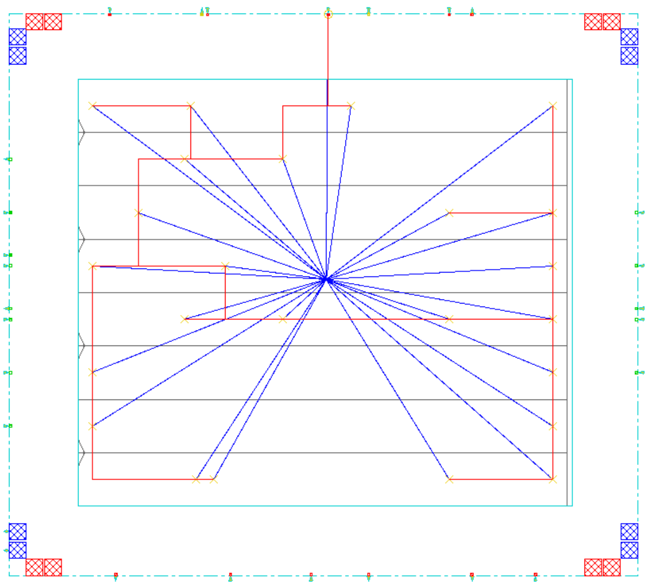
To display the generated clock tree, select
Clock -> Display -> Display Clock Tree ...
in the main menu.



[✓ All Clock(s)]
Selected Clock

[✓ Pre-Route]
Clock Route Only
Post-Route

[✓ Display Clock Tree]
[✓ All Level]
Bottom Level (non-gated clock tree only)
Selected Level (non-gated clock tree only)
1
Display Clock Phase Delay
Display Min/Max Paths

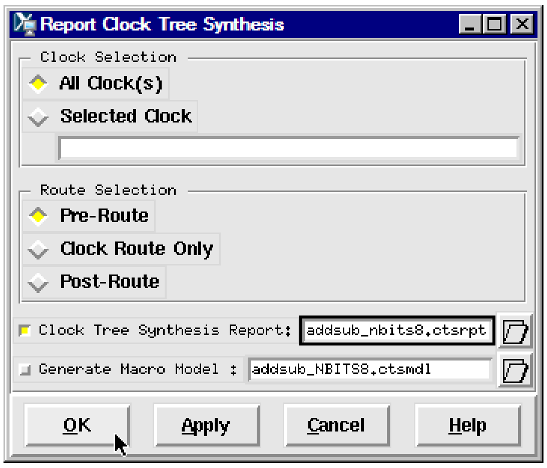


The clock paths have been balanced according to the clock tree specifications.

To get a report on the clock tree synthesis, select **Clock -> Report Clock Tree...** in the main menu.

Specify the file PAR/RPT/addsub_nbits8.ctsrpt as the report file.

Click **OK**.



Clock Selection: [✓ All Clock(s)] Selected Clock

Route Selection: [✓ Pre-Route] Clock Route Only Post-Route

[✓] Clock Tree Synthesis Report: addsub_nbits8.ctsrpt

Generate Macro Model : addsub_NBITS8.ctsmdl

The following report is also displayed in the Encounter console:

```
reportClockTree Option :
-report PAR/RPT/addsub_nbits8.ctsrpt

**** Clock Tree clk Stat ****
Total Clock Level       : 1
***** Top Nodes *****
clk delay[0(ps) 0(ps)] (  a_reg_reg_7_/C  a_reg_reg_6_/C  a_reg_reg_5_/C
a_reg_reg_4_/C  a_reg_reg_3_/C  a_reg_reg_2_/C  a_reg_reg_1_/C  a_reg_reg_0_/
C  b_reg_reg_7_/C  b_reg_reg_6_/C  b_reg_reg_5_/C  b_reg_reg_4_/C
b_reg_reg_3_/C  b_reg_reg_2_/C  b_reg_reg_1_/C  b_reg_reg_0_/C  z_reg_0_/C
z_reg_1_/C  z_reg_2_/C  z_reg_3_/C  z_reg_4_/C  z_reg_5_/C  z_reg_6_/C
z_reg_7_/C )
Level 1 (Total=24       Sink=24)
Total Sinks             : 24

********** Clock clk Pre-Route Timing Analysis **********
Nr. of Subtrees                : 1
Nr. of Sinks                   : 24
Nr. of Buffer                  : 0
Nr. of Level (including gates) : 0
Max trig. edge delay at sink(R): a_reg_reg_7_/C 9.6(ps)
Min trig. edge delay at sink(R): a_reg_reg_4_/C 0.7(ps)


                                (Actual)               (Required)
Rise Phase Delay               : 0.7~9.6(ps)           0~10000(ps)
Fall Phase Delay               : 0.7~9.6(ps)           0~10000(ps)
Trig. Edge Skew                : 8.9(ps)               300(ps)
Rise Skew                      : 8.9(ps)
Fall Skew                      : 8.9(ps)
Max. Rise Buffer Tran.         : 0(ps)                 400(ps)
Max. Fall Buffer Tran.         : 0(ps)                 400(ps)
Max. Rise Sink Tran.           : 15.5(ps)              400(ps)
Max. Fall Sink Tran.           : 15.5(ps)              400(ps)
Min. Rise Buffer Tran.         : 0(ps)                 0(ps)
Min. Fall Buffer Tran.         : 0(ps)                 0(ps)
Min. Rise Sink Tran.           : 6.5(ps)               0(ps)
Min. Fall Sink Tran.           : 6.5(ps)               0(ps)
```

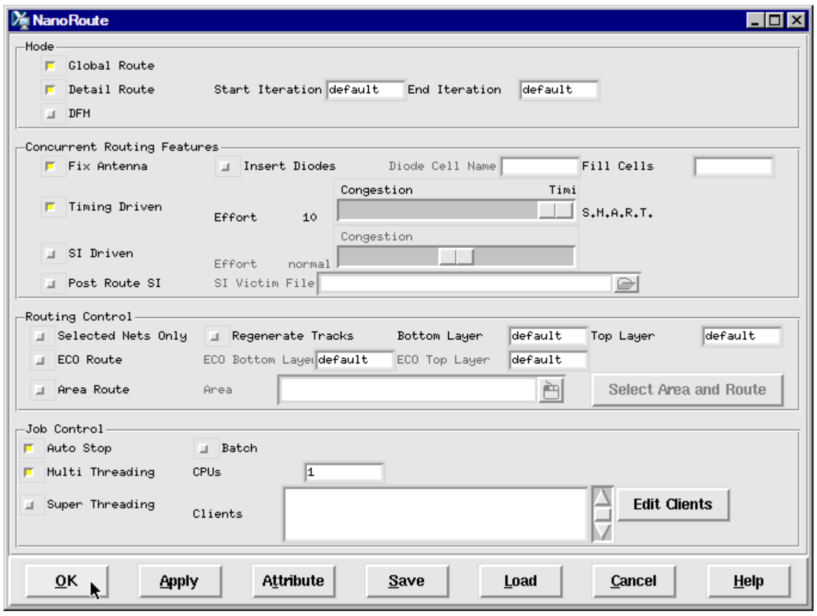
Several clock report files are also available in the PAR/CTS directory.

It is recommended to save the new stage of the design. Select **Design -> Save Design...** in the main menu and save the current state in the file PAR/DB/addsub_nbits8-cts.enc.

4.11 Design routing

This steps generates all the wires required to connect the cells according to the imported gate-level netlist.

To route the design, select **Route -> Nanoroute...** in the main menu, check the Timing Driven box and a maximum effort. Click **OK** to do the routing.


NanoRoute

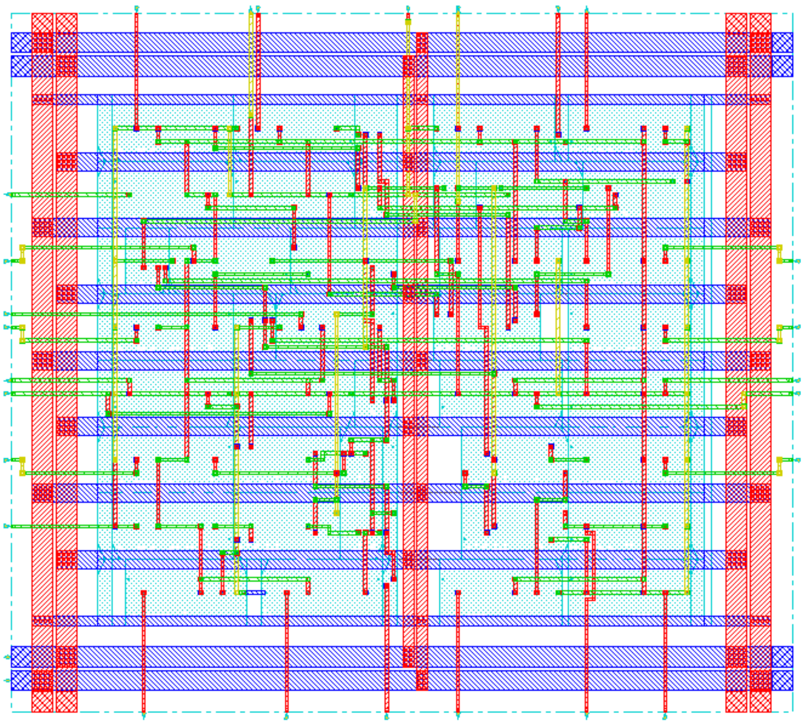
Mode
[✓ Global Route]
[✓ Detail Route] Start Iteration default End Iteration default
DFM

Concurrent Routing Features
[✓ Fix Antenna] Insert Diodes Diode Cell Name Fill Cells
[✓ Timing Driven] Congestion Effort 10 Timi S.M.A.R.T.
SI Driven Congestion Effort normal
Post Route SI SI Victim File

Routing Control
Selected Nets Only Regenerate Tracks Bottom Layer default Top Layer default
ECO Route ECO Bottom Layer default ECO Top Layer default
Area Route Area Select Area and Route

Job Control
[✓ Auto Stop] Batch
[✓ Multi Threading] CPUs 1
Super Threading Clients Edit Clients

You now get the routed design:



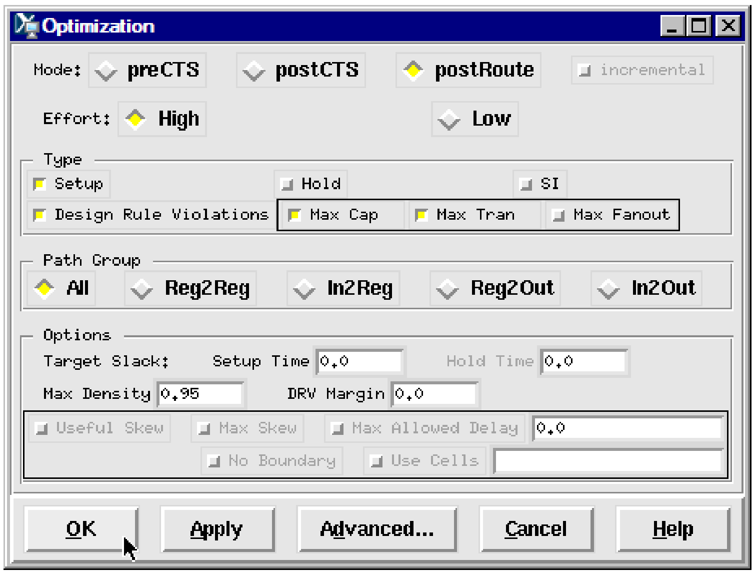
It is recommended to save the new stage of the design. Select **Design -> Save Design...** in the main menu and save the current state in the file PAR/DB/addsub_nbits8-routed.enc.

4.12 Post-routing timing optimization and analysis

A final timing optimization may be done on the routed design. Select **Timing -> Optimization...** in the main menu.

Select the **postRoute** box.

Click **OK**.


Mode: preCTS postCTS [✓ postRoute] incremental
Effort: [✓ High] Low
Type: [✓ Setup] Hold SI
[✓ Design Rule Violations] [✓ Max Cap] [✓ Max Tran] Max Fanout
Path Group: [✓ All] Reg2Reg In2Reg Reg2Out In2Out
Options: Target Slack: Setup Time 0.0 Hold Time 0.0
Max Density 0.95 DRV Margin 0.0
Useful Skew Max Skew Max Allowed Delay 0.0
No Boundary Use Cells

The results of the optimization is displayed in the Encounter console:

```
Setup mode
    Worst Slack: 2.341ns
   TNS: 0.000ns Violating
Paths: 0
    Pathgroup Slacks
        reg2reg: 2.341ns
        in2reg: 2.681ns
        reg2out: 8.959ns
        in2out: 0.000ns
Density: 96.053%
Real DRV (fanout, cap, tran): (0, 0, 0)
Total DRV (fanout, cap, tran): (0, 0, 0)
```

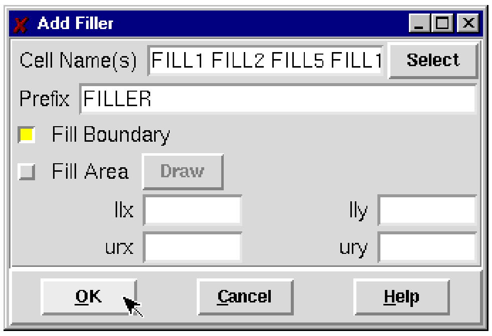
The worst slack has been improved a bit to 2.341 ns.

4.13 Filler cell placement

Filler cells will fill remaining holes in the rows and ensure the continuity of power/ground rails and N+/P+ wells in the rows.

To fill the holes with filler cells, select **Place -> Filler -> Add Filler...** in the main menu.

Select the cells FILLRT1, FILLRT2, FILLRT5, FILLRT10, FILLRT25 FILL1, FILL2, FILL5, FILL10, and FILL25 and click **OK** to place the filler cells.

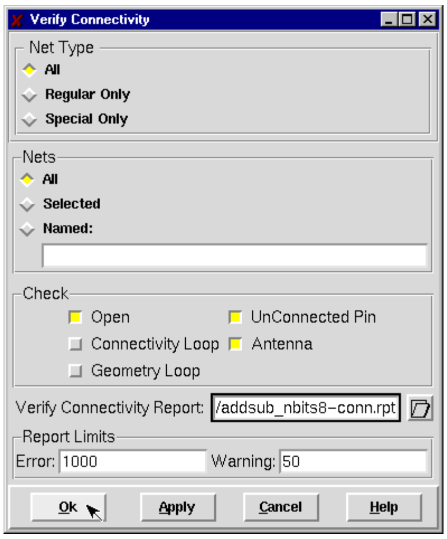

Cell Name(s) FILL1 FILL2 FILL5 FILL1
Prefix FILLER
[✓ Fill Boundary]
Fill Area
llx lly
urx ury

4.14 Design checks

The Verify menu has a number of items to check that the design has been properly placed and routed.

Select **Verify -> Verify Connectivity...** in the main menu. Define the report file as PAR/RPT/addsub_nbits8-conn.rpt.

Click **OK**. The console displays the results:


Verify Connectivity

Net Type: [✓ All] Regular Only / Special Only

Nets: [✓ All] Selected / Named:

Check:

✓ Open	✓ UnConnected Pin
☐ Connectivity Loop	✓ Antenna
☐ Geometry Loop	

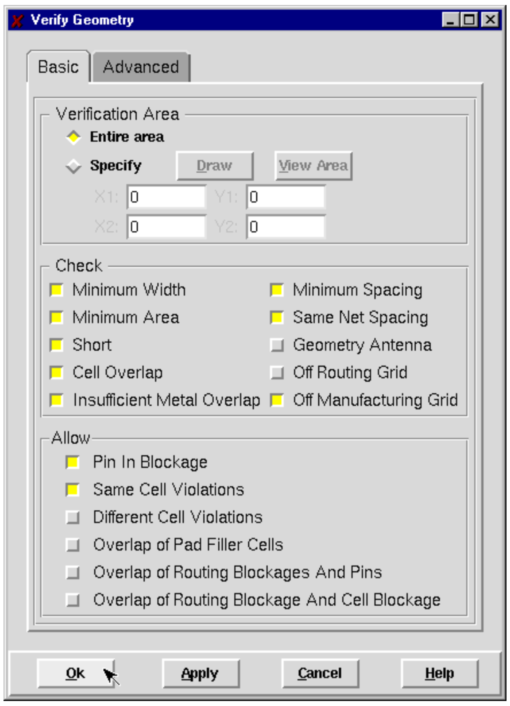
Verify Connectivity Report: /addsub_nbits8-conn.rpt

Report Limits: Error: 1000 Warning: 50

```
******** Start: VERIFY CONNECTIVITY ********
Start Time: Thu Dec  1 18:52:00 2005

Design Name: addsub_NBITS8
Database Units: 1000
Design Boundary: (0.0000, 0.0000) (153.0500,
136.9000)
Error Limit = 1000; Warning Limit = 50
Check all nets

Begin Summary
  Found no problems or warnings.
End Summary

End Time: Thu Dec  1 18:52:00 2005
******** End: VERIFY CONNECTIVITY ********
  Verification Complete : 0 Viols.  0 Wrngs.
```

Select **Verify -> Verify Geometry...** in the main menu. In the Advanced tab, define the report file as PAR/RPT/addsub_nbits8-geom.rpt.

Click **OK**.


Verify Geometry

Basic | Advanced

Verification Area: [✓ Entire area] Specify

X1: 0 Y1: 0
X2: 0 Y2: 0

Check:

✓ Minimum Width	✓ Minimum Spacing
✓ Minimum Area	✓ Same Net Spacing
✓ Short	☐ Geometry Antenna
✓ Cell Overlap	☐ Off Routing Grid
✓ Insufficient Metal Overlap	✓ Off Manufacturing Grid

Allow:

- ✓ Pin In Blockage
- ✓ Same Cell Violations
- ☐ Different Cell Violations
- ☐ Overlap of Pad Filler Cells
- ☐ Overlap of Routing Blockages And Pins
- ☐ Overlap of Routing Blockage And Cell Blockage

The console displays the results:

```
*** Starting Verify Geometry (MEM: 222.2)

VERIFY GEOMETRY ...... Starting Verification
VERIFY GEOMETRY ...... Initializing
VERIFY GEOMETRY ...... Deleting Existing
Violations
VERIFY GEOMETRY ...... Creating Sub-Areas
VERIFY GEOMETRY ...... SubArea : 1 of 1
VERIFY GEOMETRY ...... Cells    :  0 Viols.
VERIFY GEOMETRY ...... SameNet  :  0 Viols.
VERIFY GEOMETRY ...... Wiring   :  0 Viols.
VERIFY GEOMETRY ...... Antenna  :  0 Viols.
VERIFY GEOMETRY ...... Sub-Area: 1 complete 0 Viols. 0 Wrngs.
Begin Summary ...
  Cells        : 0
  SameNet      : 0
  Wiring       : 0
  Antenna      : 0
  Short        : 0
  Overlap      : 0
End Summary

  Verification Complete : 0 Viols.  0 Wrngs.
```

4.15 Report generation

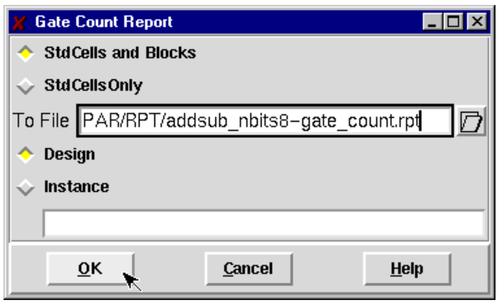
A number of reports have been already generated in the previous steps. They should be located in the PAR/RPT directory. The **Tools** menu includes some additional reports:

Tools -> Netlist Stats gives the following output in the console:

```
*** Statistics for net list addsub_NBITS8 ***
Number of cells       = 83
Number of nets        = 70
Number of tri-nets    = 0
Number of degen nets = 1
Number of pins        = 192
Number of i/os        = 27

Number of nets with    1 terms = 1 (1.4%)
Number of nets with    2 terms = 65 (92.9%)
Number of nets with    3 terms = 1 (1.4%)
Number of nets with    9 terms = 1 (1.4%)
Number of nets with >=10 terms = 2 (2.9%)

*** 10 Primitives used:
Primitive ENDCAPR (8 insts)
Primitive ENDCAPL (8 insts)
Primitive TIE0 (1 insts)
Primitive FILLRT1 (6 insts)
Primitive FILLRT2 (9 insts)
Primitive FILLRT5 (8 insts)
Primitive XNR20 (8 insts)
Primitive DFC3 (24 insts)
Primitive ADD32 (8 insts)
Primitive INV3 (3 insts)
************
```

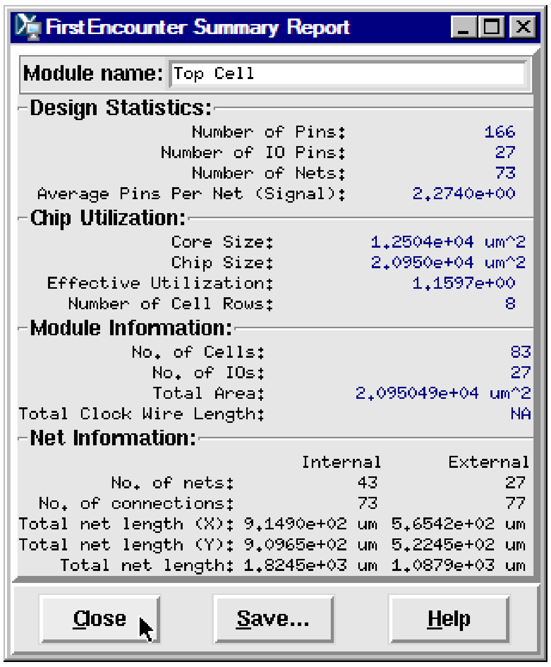
Tools -> Gate Count Report... gives the following output in the console:

```
Gate area 54.6000 um^2
```



[✓ StdCells and Blocks]
StdCells Only
To File PAR/RPT/addsub_nbits8-gate_count.rpt
[✓ Design]
Instance

Finally, **Tools -> Summary Report...** displays the following window:



```
Module name: Top Cell
Design Statistics:
                  Number of Pins:         166
               Number of IO Pins:          27
                  Number of Nets:          73
   Average Pins Per Net (Signal):  2.2740e+00
Chip Utilization:
               Core Size:    1.2504e+04 um^2
               Chip Size:    2.0950e+04 um^2
   Effective Utilization:     1.1597e+00
     Number of Cell Rows:              8
Module Information:
            No. of Cells:                 83
              No. of IOs:                 27
             Total Area:   2.095049e+04 um^2
Total Clock Wire Length:                  NA
Net Information:
                         Internal       External
          No. of nets:         43             27
   No. of connections:         73             77
Total net length (X): 9.1490e+02 um  5.6542e+02 um
Total net length (Y): 9.0965e+02 um  5.2245e+02 um
    Total net length: 1.8245e+03 um  1.0879e+03 um
```

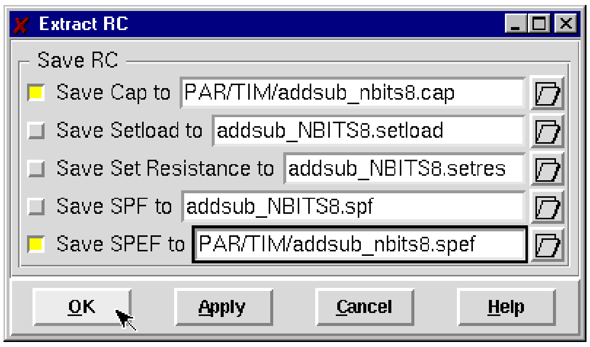
4.16 Post-route timing data extraction

This step generates the post-route SDF file that includes both the actual interconnect and cell timing delays.

The parasitics must be first extracted.
Select **Timing -> Extract RC...** in the main menu.

The generated Cap file includes the wired capacitance, pin capacitance, total capacitance, net length, wire cap per unit length and the fanout of each net in the design.

The generated SPEF (Standard Parasitics Exchange Format) file includes RC values in a SPICE-like format.

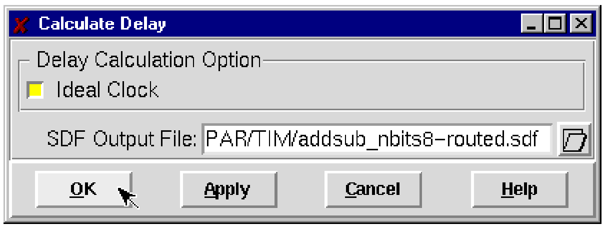


Save RC

[✓]	Save Cap to	PAR/TIM/addsub_nbits8.cap
[ ]	Save Setload to	addsub_NBITS8.setload
[ ]	Save Set Resistance to	addsub_NBITS8.setres
[ ]	Save SPF to	addsub_NBITS8.spf
[✓]	Save SPEF to	PAR/TIM/addsub_nbits8.spef

The SDF file may be then generated by selecting **Timing -> Calculate Delay...** in the main menu.

The checked Ideal Clock switch means that flip-flops are considered as having 0ps rising and falling transition times.



Delay Calculation Option
[✓] Ideal Clock
SDF Output File: PAR/TIM/addsub_nbits8-routed.sdf

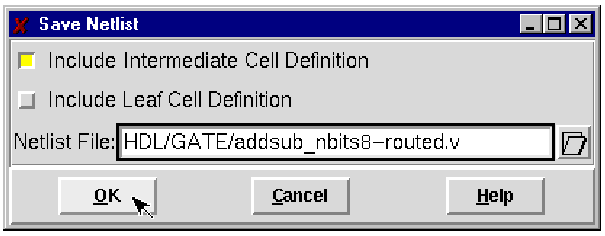
4.17 Post-route netlist generation

This steps generates a Verilog netlist of the routed design. The netlist may be different from the imported netlist as cells may have been added or replaced during clock tree synthesis and timing-driven optimizations.

Select **Design -> Save -> Netlist...** in the main menu.

Do not select Include Leaf Cell Definition as they are provided in a separate library.

The generated file should go into the HDL/GATE directory.



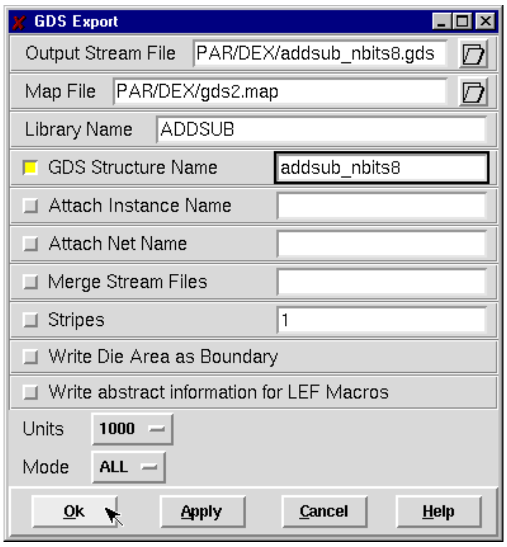
[✓] Include Intermediate Cell Definition
[ ] Include Leaf Cell Definition
Netlist File: HDL/GATE/addsub_nbits8-routed.v

4.18 GDS2 file generation

The placed and routed design can be exported in different formats for further processing outside the Encounter tool. The GDS2 binary format is a standard format for integrating the block in the top-level layout, doing DRC/LVS checkings, or delivering the layout to the foundry.

To export the design in the GDS2 format, select **Design -> Save -> GDS...** in the main menu.

The GDS map file has been copied by the AMS setup script in the PAR/DEX directory. The generated GDS2 file is written in the same directory.



Output Stream File	PAR/DEX/addsub_nbits8.gds
Map File	PAR/DEX/gds2.map
Library Name	ADDSUB
[✓] GDS Structure Name	addsub_nbits8
[ ] Attach Instance Name	
[ ] Attach Net Name	
[ ] Merge Stream Files	
[ ] Stripes	1
[ ] Write Die Area as Boundary	
[ ] Write abstract information for LEF Macros	
Units	1000
Mode	ALL

4.19 Design import in Virtuoso

The GDS2 file can be imported in Virtuoso for further processing. To do that, it is recommended to start the Cadence IC environment in the LAY directory.

Before starting Cadence IC, make sure that the following link exists:

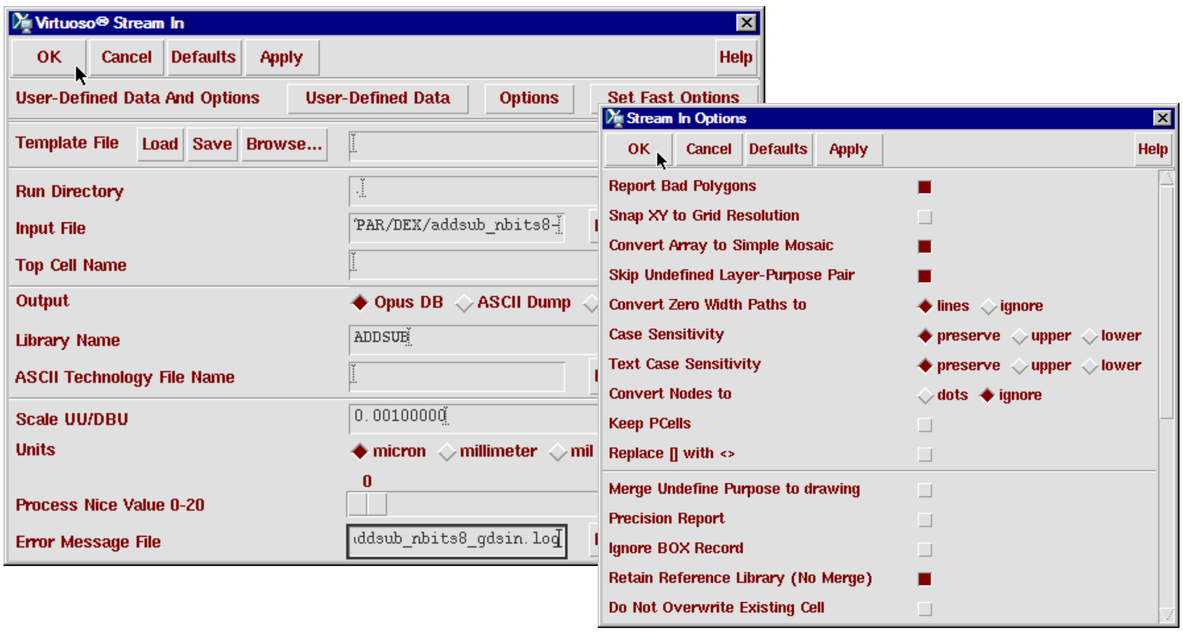
```
    [6]vachoux@immsunsrv1-LAY> ln -s ../edadk.conf .
```

Start Cadence IC:

```
    ams_cds -t c35b4 -m fb
```

Create a new library called ADDSUB with the attached technology file TECH_C35B4.

In the DFII shell, select **File -> Import -> Stream...**. In the Options tab, check the Retain Reference Library (No Merge) option. Since the cell layouts are available in the CORELIB library, it is possible to display the full layout of the block.

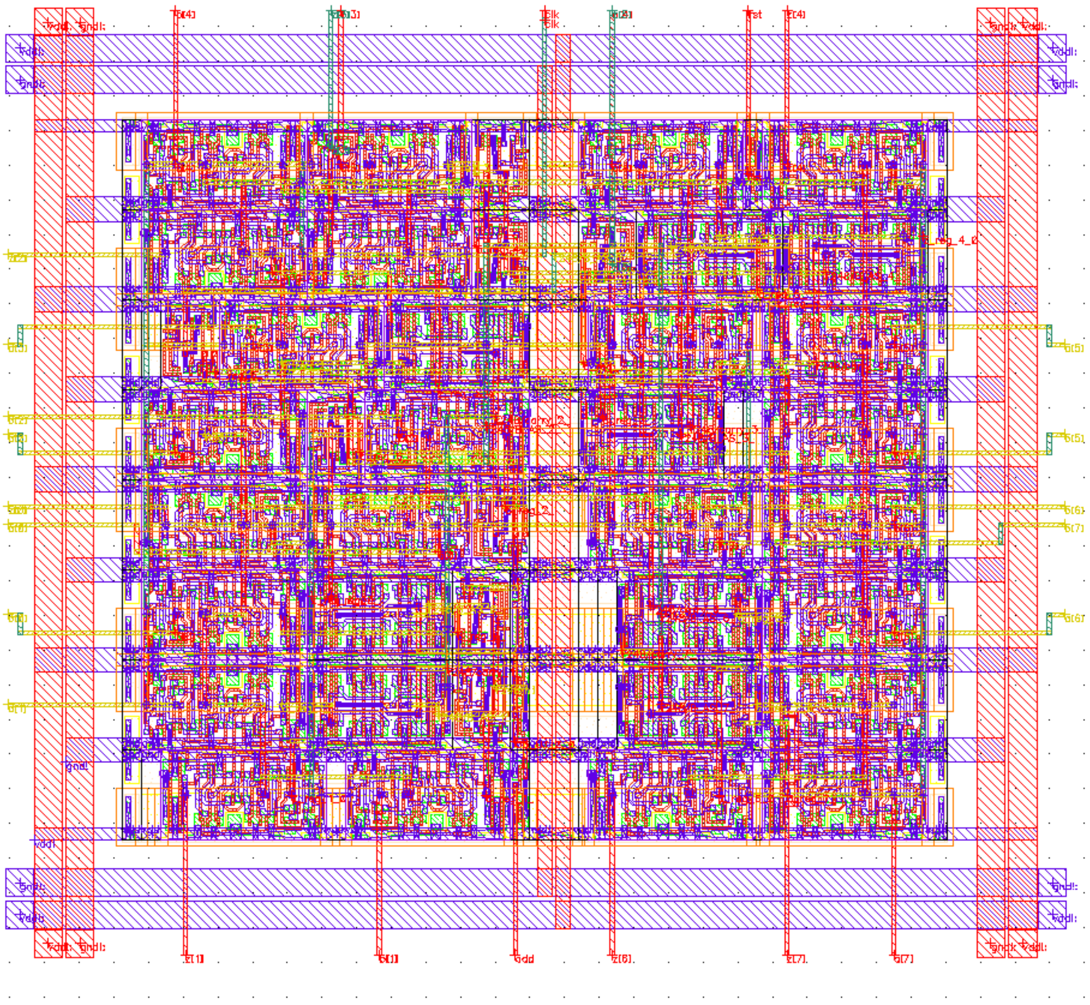


Template File	
Run Directory	.
Input File	PAR/DEX/addsub_nbits8-
Top Cell Name	
Output	[✓ Opus DB] ASCII Dump
Library Name	ADDSUB
ASCII Technology File Name	
Scale UU/DBU	0.00100000
Units	[✓ micron] millimeter mil
Process Nice Value 0-20	0
Error Message File	ddsub_nbits8_gdsin.log

Report Bad Polygons	✓
Snap XY to Grid Resolution	☐
Convert Array to Simple Mosaic	✓
Skip Undefined Layer-Purpose Pair	✓
Convert Zero Width Paths to	[✓ lines] ignore
Case Sensitivity	[✓ preserve] upper lower
Text Case Sensitivity	[✓ preserve] upper lower
Convert Nodes to	dots [✓ ignore]
Keep PCells	☐
Replace [] with <>	☐
Merge Undefine Purpose to drawing	☐
Precision Report	☐
Ignore BOX Record	☐
Retain Reference Library (No Merge)	✓
Do Not Overwrite Existing Cell	☐

A log file called addsub_nbits8_gdsin.log contains a summary of the GDS2 import.

The full imported layout is given below:



4.20 Using scripts

As for the synthesis step, it is much more convenient to capture the placement and routing flow in a script. Cadence Encounter also support sthe Tcl language for building scripts.

An example of such a script for placement and routing of the adder-subtractor design has been installed in the PAR/BIN directory (see "1.5 VHDL example: Adder-subtractor"). The script must be run from the project top directory and it assumes a directory organization as described in "1.2 Design project organisation".

To run the Tcl script, execute the following command in a Unix shell:

```
encounter -log PAR/LOG/encounter -overwrite -init PAR/BIN/addsub_par.tcl -win
```

The script is given below. It may be modified to define design information and to control the flow to some extent.

Note that a configuration file must exist before running the script. The configuration file name is in the PAR/CONF directory and its name is defined in the script.

The script does a bit more than the steps described earlier. For example, it uses an I/O pin placement definition as provided by a PAR/CONF/*.io file.

```
#
#   Cadence Encounter Tcl script for adder-subtractor.
#
#   Process: AMS 0.35u CMOS (C35), Hit-Kit 3.70
#

#-----------------------------------------------------------------------------
#   It is assumed that a project directory structure has already been
#   created using 'create_project' and that this synthesis script is
#   executed from the project root directory $PROJECT_DIR
#-----------------------------------------------------------------------------
set PROJECT_DIR [pwd]

#-----------------------------------------------------------------------------
#   Design related information (can be changed)
#-----------------------------------------------------------------------------
set DESIGN addsub_nbits8

set TIM_LIBRARY   C35_CORELIB
set TIM_OC_MAX    WORST       ;# TYPICAL | WORST | WORST-IND
set TIM_OC_MIN    BEST        ;# TYPICAL | BEST | BEST-IND

# Floorplan settings
#
set FP_ASPECT_RATIO  1
set FP_ROW_DENSITY   0.85     ;# percent
set FP_CORE2IO       16       ;# micron

# Power ring and settings
#
set PR_WIDTH         4        ;# micron
set PR_SPACING       1        ;# micron
set PR_LAYER_TB      MET1     ;# top and bottom layer
set PR_LAYER_LR      MET2     ;# left and right layer

# Power stripe settings
#
set ST_NUM_SETS      1        ;# number of sets
set ST_SPACING       1        ;# micron
set ST_LAYER_V       $PR_LAYER_LR
set ST_WIDTH         2        ;# micron
set ST_XOFS_R        60       ;# micron
set ST_XOFS_L        60       ;# micron

# Placement settings
#
set PL_EFFORT        -high  ;# -low | -medium | -high

# Clock tree synthesis settings
#
set CTS_BUFFER    BUF2
set CTS_INV       INV0

#-----------------------------------------------------------------------------
#   Flags that drive the script behavior (can be changed)
#
#   ADD_STRIPES (0 | 1)
#      if 1, add stripes
#   PLACE_TIMING (0 | 1)
#      if 1, do a timing driven placement
#   CLOCK_TREE (0 | 1)
#      if 1, create a clock tree
#   CTS_CREATE_SPEC (0 | 1)
#      if 1, create a clock tree specification file with default values
#   ROUTE_TIMING (0 | 1)
#      if 1, do a timing driven routing
#   OPT (string)
#      can be used to have different generated file names
```

```
#------------------------------------------------------------------------------
set ADD_STRIPES 1
set PLACE_TIMING 1
set CLOCK_TREE 1
set CTS_CREATE_SPEC 0
set ROUTE_TIMING 1
set OPT "_cts"

#------------------------------------------------------------------------------
#   File names
#------------------------------------------------------------------------------
set CONF_FILE_NAME          ${DESIGN}.conf
set IO_FILE_NAME            ${DESIGN}.io
set DESIGN_NAME             ${DESIGN}${OPT}
set SAVE_DESIGN_FP_NAME     ${DESIGN_NAME}-fplan.enc
set SAVE_DESIGN_PR_NAME     ${DESIGN_NAME}-pring.enc
set SAVE_DESIGN_PL_NAME     ${DESIGN_NAME}-placed.enc
set SAVE_DESIGN_PF_NAME     ${DESIGN_NAME}-placed_filled.enc
set SAVE_DESIGN_CT_NAME     ${DESIGN_NAME}-cts.enc
set SAVE_DESIGN_RO_NAME     ${DESIGN_NAME}-routed.enc
set TIM_RCDB_NAME           ${DESIGN_NAME}.rcdb
set SDF_FILE_NAME           ${DESIGN_NAME}-routed.sdf
set SPEF_FILE_NAME          ${DESIGN_NAME}-routed.spef
set RPT_CHECK_TA_NAME       ${DESIGN_NAME}-checkta.rpt
set RPT_REPORT_TA_NAME      ${DESIGN_NAME}-ta.rpt
set RPT_SLACK_NAME          ${DESIGN_NAME}-slack.rpt
set RPT_GATE_COUNT_NAME     ${DESIGN_NAME}-gate_count.rpt
set RPT_NOTCH_NAME          ${DESIGN_NAME}-notch.rpt
set RPT_CONN_NAME           ${DESIGN_NAME}-conn.rpt
set RPT_GEOM_NAME           ${DESIGN_NAME}-geom.rpt
set RPT_DENSITY_NAME        ${DESIGN_NAME}-density.rpt
set VLOG_NETLIST_SIM_NAME   ${DESIGN_NAME}-routed.v
set VLOG_NETLIST_LVS_NAME   ${DESIGN_NAME}-routed_lvs.v
set CTS_SPEC_NAME           ${DESIGN_NAME}-spec.cts
set CTS_RGUIDE_NAME         ${DESIGN_NAME}-guide.cts
set CTS_RPT_NAME            ${DESIGN_NAME}-cts.rpt
set GDS_FILE_NAME           ${DESIGN_NAME}.gds

#------------------------------------------------------------------------------
#   Absolute paths
#------------------------------------------------------------------------------
set CONF_FILE               ${PROJECT_DIR}/PAR/CONF/${CONF_FILE_NAME}
set IO_FILE                 ${PROJECT_DIR}/PAR/CONF/${IO_FILE_NAME}
set SAVE_DESIGN_FP_FILE     ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_FP_NAME}
set SAVE_DESIGN_PR_FILE     ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_PR_NAME}
set SAVE_DESIGN_PL_FILE     ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_PL_NAME}
set SAVE_DESIGN_PF_FILE     ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_PF_NAME}
set SAVE_DESIGN_CT_FILE     ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_CT_NAME}
set SAVE_DESIGN_RO_FILE     ${PROJECT_DIR}/PAR/DB/${SAVE_DESIGN_RO_NAME}
set SDF_FILE                ${PROJECT_DIR}/PAR/TIM/${SDF_FILE_NAME}
set SPEF_FILE               ${PROJECT_DIR}/PAR/TIM/${SPEF_FILE_NAME}
set TIM_RCDB_FILE           ${PROJECT_DIR}/PAR/TIM/${TIM_RCDB_NAME}
set RPT_CHECK_TA_FILE       ${PROJECT_DIR}/PAR/RPT/${RPT_CHECK_TA_NAME}
set RPT_REPORT_TA_FILE      ${PROJECT_DIR}/PAR/RPT/${RPT_REPORT_TA_NAME}
set RPT_SLACK_FILE          ${PROJECT_DIR}/PAR/RPT/${RPT_SLACK_NAME}
set RPT_GATE_COUNT_FILE     ${PROJECT_DIR}/PAR/RPT/${RPT_GATE_COUNT_NAME}
set RPT_NOTCH_FILE          ${PROJECT_DIR}/PAR/RPT/${RPT_NOTCH_NAME}
set RPT_CONN_FILE           ${PROJECT_DIR}/PAR/RPT/${RPT_CONN_NAME}
set RPT_GEOM_FILE           ${PROJECT_DIR}/PAR/RPT/${RPT_GEOM_NAME}
set RPT_DENSITY_FILE        ${PROJECT_DIR}/PAR/RPT/${RPT_DENSITY_NAME}
set VLOG_NETLIST_SIM_FILE   ${PROJECT_DIR}/HDL/GATE/${VLOG_NETLIST_SIM_NAME}
set VLOG_NETLIST_LVS_FILE   ${PROJECT_DIR}/HDL/GATE/${VLOG_NETLIST_LVS_NAME}
set CTS_SPEC_FILE           ${PROJECT_DIR}/PAR/CTS/${CTS_SPEC_NAME}
set CTS_RGUIDE_FILE         ${PROJECT_DIR}/PAR/CTS/${CTS_RGUIDE_NAME}
set CTS_RPT_FILE            ${PROJECT_DIR}/PAR/RPT/${CTS_RPT_NAME}
set GDS_FILE                ${PROJECT_DIR}/PAR/DEX/${GDS_FILE_NAME}
set GDS_MAP_FILE            ${PROJECT_DIR}/PAR/DEX/gds2.map
```

```
#-----------------------------------------------------------------------------
#   Procedures
#-----------------------------------------------------------------------------

# make_clock_tree
#
proc make_clock_tree create_spec {

   global CTS_BUFFER CTS_INV CTS_SPEC_FILE CTS_RGUIDE_FILE CTS_RPT_FILE

   if { $create_spec || ![file exists $CTS_SPEC_FILE] } {
      createClockTreeSpec \
         -bufFootprint $CTS_BUFFER \
         -invFootprint $CTS_INV \
         -output $CTS_SPEC_FILE
   }
   specifyClockTree -clkfile $CTS_SPEC_FILE
   ckSynthesis \
      -rguide $CTS_RGUIDE_FILE \
      -report $CTS_RPT_FILE
   optDesign -postCTS -setup -drv -outDir PAR/RPT

} ;# make_clock_tree

#-----------------------------------------------------------------------------
#   Load configuration file
#-----------------------------------------------------------------------------
loadConfig $CONF_FILE 0
commitConfig

#-----------------------------------------------------------------------------
#   Load IO file
#-----------------------------------------------------------------------------
loadIoFile $IO_FILE

#-----------------------------------------------------------------------------
#   Set operating conditions
#-----------------------------------------------------------------------------
setOpCond \
   -maxLibrary $TIM_LIBRARY -max $TIM_OC_MAX \
   -minLibrary $TIM_LIBRARY -min $TIM_OC_MIN

#-----------------------------------------------------------------------------
#   Set user grids
#-----------------------------------------------------------------------------
setPreference ConstraintUserXGrid 0.1
setPreference ConstraintUserXOffset 0.1
setPreference ConstraintUserYGrid 0.1
setPreference ConstraintUserYOffset 0.1
setPreference SnapAllCorners 1
setPreference BlockSnapRule 2

#-----------------------------------------------------------------------------
#   Define global Power nets - make global connections
#-----------------------------------------------------------------------------
clearGlobalNets
globalNetConnect vdd! -type pgpin -pin vdd! -inst * -module {} -verbose
globalNetConnect gnd! -type pgpin -pin gnd! -inst * -module {} -verbose
#globalNetConnect vdd3o! -type pgpin -pin vdd3o! -inst * -module {} -verbose
#globalNetConnect vdd3r1! -type pgpin -pin vdd3r1! -inst * -module {} -verbose
#globalNetConnect vdd3r2! -type pgpin -pin vdd3r2! -inst * -module {} -verbose
#globalNetConnect gnd3o! -type pgpin -pin gnd3o! -inst * -module {} -verbose
#globalNetConnect gnd3r! -type pgpin -pin gnd3r! -inst * -module {} -verbose

#-----------------------------------------------------------------------------
#   Initialize floorplan
#-----------------------------------------------------------------------------
floorPlan -r $FP_ASPECT_RATIO \
   $FP_ROW_DENSITY \
   $FP_CORE2IO $FP_CORE2IO $FP_CORE2IO $FP_CORE2IO
fit
saveDesign $SAVE_DESIGN_FP_FILE

#-----------------------------------------------------------------------------
#   Add CAP cells
#-----------------------------------------------------------------------------
addEndCap -preCap ENDCAPL -postCap ENDCAPR -prefix ENDCAP

#-----------------------------------------------------------------------------
#   Create and route power rings and power stripes
#-----------------------------------------------------------------------------
addRing \
   -around core \
   -nets { gnd! vdd! } \
   -width_bottom $PR_WIDTH -width_top $PR_WIDTH \
   -width_left $PR_WIDTH -width_right $PR_WIDTH \
   -spacing_bottom $PR_SPACING -spacing_top $PR_SPACING \
   -spacing_left $PR_SPACING -spacing_right $PR_SPACING \
   -layer_bottom $PR_LAYER_TB -layer_top $PR_LAYER_TB \
   -layer_left $PR_LAYER_LR -layer_right $PR_LAYER_LR \
   -center 1 \
   -tl 1 -tr 1 -bl 1 -br 1 -lt 1 -lb 1 -rt 1 -rb 1 \
   -stacked_via_bottom_layer MET1 -stacked_via_top_layer MET4 \
   -threshold 0.7
if { $ADD_STRIPES } {
   addStripe \
      -nets { gnd! vdd! } \
      -number_of_sets $ST_NUM_SETS \
      -spacing $ST_SPACING \
      -layer $ST_LAYER_V \
      -width $ST_WIDTH \
      -xleft_offset $ST_XOFS_L
}
sroute \
   -jogControl { preferWithChanges differentLayer } \
   -nets { gnd! vdd! }
saveDesign $SAVE_DESIGN_PR_FILE

#-----------------------------------------------------------------------------
#   Core cell placement
#-----------------------------------------------------------------------------
if { $PLACE_TIMING } {
   amoebaPlace $PL_EFFORT -timingdriven
} else {
   amoebaPlace $PL_EFFORT
}
setDrawMode place
saveDesign $SAVE_DESIGN_PL_FILE

#-----------------------------------------------------------------------------
#   Create clock tree (optional)
#-----------------------------------------------------------------------------
if { $CLOCK_TREE } {
   make_clock_tree $CTS_CREATE_SPEC
   saveDesign $SAVE_DESIGN_CT_FILE
}

#-----------------------------------------------------------------------------
#   Add filler cells
#-----------------------------------------------------------------------------
addFiller -cell FILL25 FILL10 FILL5 FILL2 FILL1 -prefix FILLER
saveDesign $SAVE_DESIGN_PF_FILE

#-----------------------------------------------------------------------------
#   Route design (Nanoroute)
#-----------------------------------------------------------------------------
```

```
if { $ROUTE_TIMING } {
   setNanoRouteMode -quiet -timingEngine CTE
   setNanoRouteMode -quiet -routeWithTimingDriven true
   setNanoRouteMode -quiet -routeTdrEffort 0
}
globalDetailRoute
optDesign -postRoute -setup -drv -outDir PAR/RPT
saveDesign $SAVE_DESIGN_RO_FILE
setDrawMode place

#----------------------------------------------------------------------------
#   Verifications
#----------------------------------------------------------------------------
fillNotch -report $RPT_NOTCH_FILE
verifyConnectivity \
   -type all \
   -error 1000 \
   -warning 50 \
   -report $RPT_CONN_FILE
verifyGeometry \
   -allowSameCellViols \
   -allowRoutingBlkgPinOverlap \
   -allowRoutingCellBlkgOverlap \
   -report $RPT_GEOM_FILE
verifyMetalDensity \
   -detailed \
   -report $RPT_DENSITY_FILE

#----------------------------------------------------------------------------
#   Extract parasitics
#----------------------------------------------------------------------------
setExtractRCMode \
   -detail \
   -rcdb $TIM_RCDB_FILE \
   -relative_c_t 0.01 \
   -total_c_t 5.0 \
   -reduce 5
extractRC

#----------------------------------------------------------------------------
#   Generate RC and timing files
#----------------------------------------------------------------------------
rcOut -spef $SPEF_FILE
delayCal -sdf $SDF_FILE

#----------------------------------------------------------------------------
#   Generate reports
#----------------------------------------------------------------------------
reportGateCount -outfile $RPT_GATE_COUNT_FILE

# Timings
#
setCteReport
setAnalysisMode -setup -async -skew -noClockTree -sequentialConstProp
reportAnalysisMode
buildTimingGraph
checkTA -verbose > $RPT_CHECK_TA_FILE
reportTA \
   -format { hpin arc cell delay arrival required slew fanout load } \
   -late \
   -max_points 10 \
   -net \
   > $RPT_REPORT_TA_FILE

#----------------------------------------------------------------------------
#   Save netlist
#----------------------------------------------------------------------------
saveNetlist -excludeLeafCell $VLOG_NETLIST_SIM_FILE
saveNetlist -physical $VLOG_NETLIST_LVS_FILE

#-----------------------------------------------------------------------------
#   Save GDS2
#-----------------------------------------------------------------------------
streamOut $GDS_FILE \
   -mapFile $GDS_MAP_FILE \
   -libName ADDSUB \
   -structureName $DESIGN_NAME \
   -stripes $ST_NUM_SETS \
   -units 1000 \
   -mode ALL
```