

Verification Continuum™

# **HAPS® Prototyping**

## **Reference Manual**

---

April 2022

**SYNOPSYS®**

[solvnetplus.synopsys.com](https://solvnetplus.synopsys.com)

Synopsys Confidential Information

## Copyright Notice and Proprietary Information

© 2022 Synopsys, Inc. This Synopsys software and all associated documentation are proprietary to Synopsys, Inc. and may only be used pursuant to the terms and conditions of a written license agreement with Synopsys, Inc. All other use, reproduction, modification, or distribution of the Synopsys software or the associated documentation is strictly prohibited.

## Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

## Destination Control Statement

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

## Disclaimer

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

## Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at <http://www.synopsys.com/Company/Pages/Trademarks.aspx>. All other product or company names may be trademarks of their respective owners.

## **Third-Party Links**

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.  
690 East Middlefield Road  
Mountain View, CA 94043  
[www.synopsys.com](http://www.synopsys.com)

April 2022

## **Synopsys Statement on Inclusivity and Diversity**

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

# Contents

---

## **Chapter 1: Design Basics**

Design Processes .....	10
Design Database Structure .....	11

## **Chapter 2: Design Flow Scripts**

Synthesis Flow .....	18
FPGA Synthesis with Multiple Task Exploration .....	20
Debug Scripts .....	22

## **Chapter 3: Graphical User Interface**

Graphical Interface .....	26
Graphical Window .....	29
Database View .....	29
Tcl Script Window .....	30
TSS Builder Window .....	30
Menus .....	32
View Menu .....	32
Database Menu .....	33
State Menu .....	36
Tools Menu .....	39
Timing Report View .....	40
Timing Analyst .....	48
TSS Builder .....	57
Options Menu .....	70
Tech-Support Menu .....	81
Web Menu .....	82
Help Menu .....	83
Icons .....	85
State Tree View Operations .....	88

State Tree Bottom Menu	90
Design Flow View Operations	91
Partition Design Flow	91
Synthesis Design Flow	93
Design Flow Bottom Menu	94
Design Flow Popup Menu	94
Secondary Views	96
Database Files View	96
Summary View	97
Report Messages View	97
Task Dialog Boxes	100
Run Compile Dialog Box	101
Run Instrumentation Preparation Dialog Box	103
Run Pre-Map Dialog Box	105
Run Map Dialog Box	107
Run Pre-Partition Dialog Box	108
Run Partition Dialog Box	109
Run System Route Dialog Box	111
Run System Generate Dialog Box	113
PCF Graphical Editor	116
PCF Graphical Editor Window	116
Ports Tab	119
Global Nets Tab	121
Cells Tab	123
Tcl Editor Tab	126
File List Editor	128
File List Editor (Detail)	128
Find in Files Dialog Box	130
Select Files Dialog Box	131
Options Editor	134
Add Vivado IP	136
Verdi Debug Setup	137

## Chapter 4: Feature Descriptions

Unified Compile Support	142
UPF Support	144
Time Budgeting	146

Time Budgeting Between Partitions .....	146
Time Budgeting Calculation .....	152

## Chapter 5: Inputs, Reports and Log Files

Report and Log File Access .....	156
Report View .....	157
Database Reports and Log Files .....	158
List of Report and Log Files .....	159
Synthesis Database Report and Log Files .....	160
Pre-Partition Database State Log Files and Reports .....	163
Partition Database State Log Files and Reports .....	165
System-Route Database State Log Files and Reports .....	168
System-Generate Database State Log Files and Reports .....	169
Miscellaneous Log Files and Reports .....	171
Configuration Commands for Partitioning Reports .....	173
UPF Report .....	183
Formal Verification Report and Files .....	186
Unified Compile Input Files .....	188
UPF File for Unified Compile .....	188
UTF File .....	188
VCS Script .....	191
Proto_rt Input Files .....	193





## CHAPTER 1

# Design Basics

---

Synopsys<sup>®</sup> HAPS<sup>®</sup> ProtoCompiler and ProtoCompiler DX are design automation and debug software environments designed expressly for prototyping and use with Synopsys HAPS board systems.

This document provides descriptions of the features available with both the ProtoCompiler and ProtoCompiler DX products. This chapter discusses the following basics for these tools:

- [Design Processes](#), on page 10
- [Design Database Structure](#), on page 11

# Design Processes

The ProtoCompiler DX package includes features for logic synthesis, timing analysis, RTL debug, and FPGA place and route automation targeted for a HAPS-DX, single FPGA board system. The ProtoCompiler package adds features to support partitioning and routing within a HAPS multi-FPGA board system. Both packages include features to ease the migration of ASIC RTL and IP as well as runtime features to enable system configuration and debug visibility into the live hardware.

After the tool is started with the appropriate startup command ([Chapter 2, \*Shell Command Reference\*](#) in the *Command Reference Manual*), the other design processes for the flow are executed through entered commands. Each tool has a command shell, where you can issue commands and run processes. The shell lets you create a design project database to use as a repository for the design tasks such as compiling and mapping for logic synthesis, debugging, place and route, and result reporting. The ProtoCompiler product additionally includes tasks for design partitioning and routing among multiple FPGAs.

Design processes are completed using a set of common commands, grouped by tasks, as listed below:

database	Sets the state or context for executing design tasks, querying task status, and saving, retrieving, and manipulating design data
run	Launches design processes such as logic synthesis and place and route
report	Reports the result status of a process
option	Specifies or reports option value settings
edit	Creates and modifies design files
export	Writes design files to a location outside of the database
view	Displays file contents

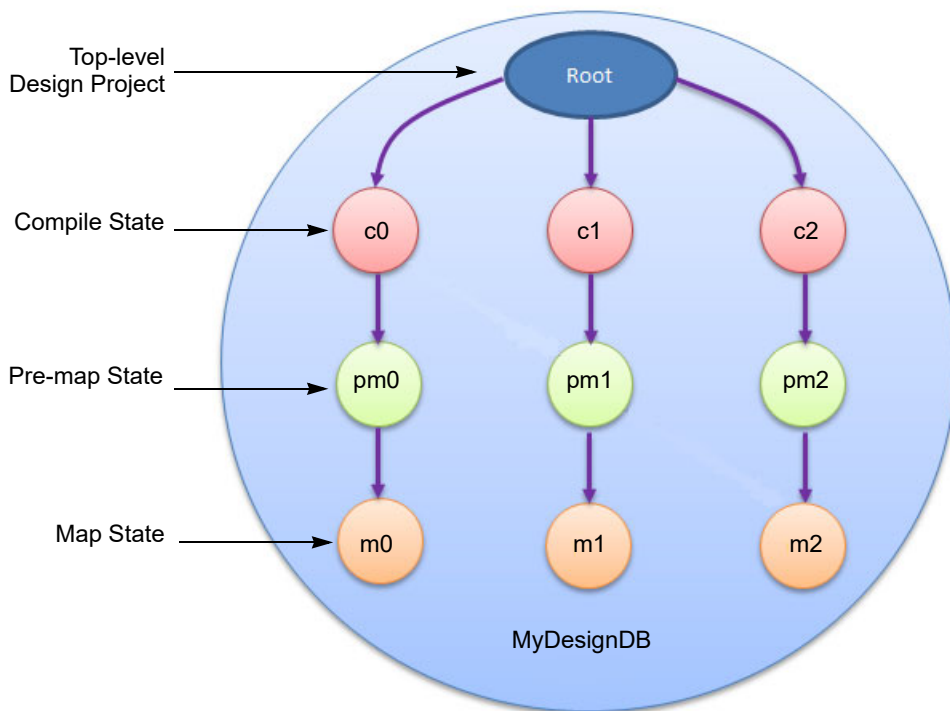
For details about these shell commands, see [Chapter 2, \*Shell Command Reference\*](#) in the *Command Reference Manual*.

# Design Database Structure

The database for the ProtoCompiler and ProtoCompiler DX products is a repository for all data associated with a design project and consists of database states created from the commands you run in the design flow. From the database, you can create, access, manipulate, and examine design files.

## The Structure

The following figure illustrates how the database is structured and how the design files for different tasks are created and stored.



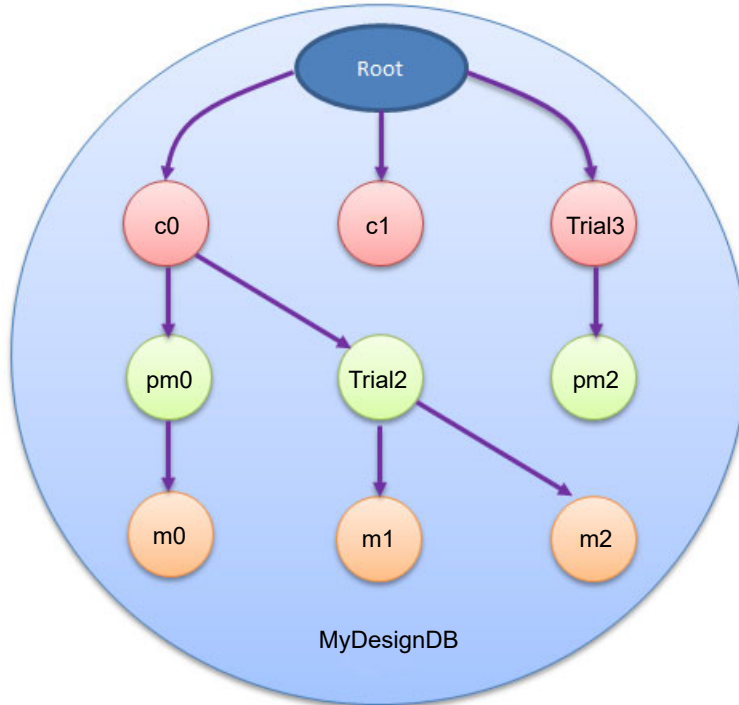
## Database States and the Design Flow

The top-level design project is referred to as the root of the database and contains only information related to the database states and the current state. Each state includes a pointer to its predecessor and pointers to all of its successors. The states are defined and manipulated by a set of shell process commands. Commands that extract information use the current state as input, and commands that create information either update the current state or create a parallel state. For step-by-step descriptions of working with databases, refer to [Working with a Design Database, on page 58](#) in the *User Guide*.

Database states follow the design flow. Running the compile process creates or modifies a compile state. The pre-map process creates a pre-map database state from information in the preceding compile state, and the map process creates a map database state from information in the pre-map state. Process commands are repeated to modify information within a database state.

A database is locked while it is being processed. You can check whether a database is locked or unlocked using the command `database query_state`.

The following figure illustrates database state manipulation within a top-level design project. The figure assumes that performance goals are not met at database states m0, m1, c1, and pm2. The table after the figure shows this command sequence.



Command	Result
<code>database create MyDesignDB</code>	Creates the root project MyDesignDB.
<code>source compile_opts0.tcl</code>	Sources the Tcl file defining the compile options.
<code>run compile -srclist fl</code>	Runs the compile process on the listed files using the defined compile options to create database state c0.
<code>source premap_opts0.tcl</code>	Sources the Tcl file defining the pre-map options.
<code>run pre_map -file my.fdc</code>	Runs the pre-map process using the defined options and the constraints from the FDC file to create database state pm0.
<code>source map_opts0.tcl</code>	Sources the Tcl file defining the map options.
<code>run map</code>	Runs the map process to create database state m0 (reject results).
<code>database set root</code>	Resets the database to root.

Command	Result
<code>source compile_opts1.tcl</code>	Sources a new Tcl file defining updated compile options.
<code>run compile -srclist f1</code>	Runs the compile process on the listed files using the updated compile options to create database state c1 (reject results).
<code>database set c0</code>	Resets the current node to c0.
<code>source pre_map_opts1.tcl</code>	Sources a new Tcl file defining pre-map options.
<code>run pre_map -file my.fdc -out Trial2</code>	Runs the pre-map process using the defined options and the constraints from the FDC file. Names the database state Trial2.
<code>source map_opts1.tcl</code>	Sources a new Tcl file defining the map options.
<code>run map</code>	Runs the map process to create database state m1 (reject results).
<code>database close</code>	Closes the database.
<code>database load MyDesignDB</code>	Reloads the root project MyDesignDB.
<code>source compile_opts2.tcl</code>	Sources a new Tcl file defining compile options.
<code>run compile -srclist f1 -out Trial3</code>	Runs the compile process on the listed files using updated compile options. Names the database state Trial3.
<code>source premap_opts2.tcl</code>	Sources a new Tcl file defining pre-map options.
<code>run pre_map -file my.fdc</code>	Runs the pre-map process using the defined options and the constraints from the FDC file to create database state pm2 (reject results).
<code>database set Trial2</code>	Sets the current database state to Trial2
<code>source map_opts2.tcl</code>	Sources a new Tcl file defining map options.
<code>run map</code>	Runs the map process to create database state m2 (accept results).

## Database File Access

The database is a closed database and you cannot manipulate the files directly. If you want to database state files, you can export individual files or packages of files for place and route, simulation, or verification (see [export](#),

[on page 54](#) in the *Command Reference Manual*). You can access or use an offline viewing mechanism to access reports generated by various states (see [Checking Reports and Log Files, on page 512](#) in the *User Guide*).

You can also archive databases or extract files from archived databases using a built-in utility. For more information, see [Archiving and Unarchiving Databases, on page 66](#) in the *User Guide*.





## CHAPTER 2

# Design Flow Scripts

---

This section provides a set of example design flow scripts, which include:

- [Synthesis Flow](#), on page 18
- [FPGA Synthesis with Multiple Task Exploration](#), on page 20
- [Debug Scripts](#), on page 22

# Synthesis Flow

The following examples illustrate commands executed in the shell to support a general design flow for synthesizing a design.

## RTL Preparation

This example shows how to prepare the RTL source files.

```
# Launch a text editor to create an RTL file list; a file with
# a list of RTL files for the design. Save the file and quit
# the text editor when done:

% edit file MyRtlFilelist

# Launch a text editor to create a compiler options file; a
# file defining the desired compiler options for the run.
# Save the file and quit the text editor when done:

% edit file MyCompilerOptionsFile

# Report RTL diagnostics to flush out all parse errors. Analyze
# the report file for errors, correct, and re-iterate:

% source MyCompilerOptions.tcl
% report syntax_check -filelist MyRtlFilelist.lst

# When the RTL is clean, proceed through full design compilation.
# The compile options set by sourcing MyCompilerOptions.tcl
# still apply:

% run compile -srclist MyRtlFilelist
```

## FDC Constraint Application

This example continues from [RTL Preparation, on page 18](#) and shows the application of FDC constraints.

```
# Launch a text editor to create a "global prep" options file: a
# file defining the desired "global prep" options for the run.
# Save the file and quit the text editor when done:

% edit file MyGlobalPrepOptions.tcl
```

```
# Create a constraints file using the constraints editor.
# Objects get pre-populated in the editor because the current
# database state is the compile state. When done in the editor,
# save the file but leave the application open:

% edit fdc -mode gui MyFdcFile.fdc

# Execute the "global preparation" operation:

% source MyGlobalPrepOptions.tcl
% run pre_map -file MyFdcFile.fdc -out MyPrep
% view report

# Examine the report file. Modify the constraints based on the
# report. Save the modified FDC file and re-apply constraints.
# Iterate with run pre_map until satisfied:
```

## Mapper Execution

This example illustrates continues the design flow from [FDC Constraint Application, on page 18](#) and illustrates the mapper process.

```
# Launch a text editor to create a mapper options file; a file
# defining the desired mapper options for the run. Save the
# file and quit the text editor when done:

% edit file MyMapperOptions.tcl

# Run the mapper:

% source MyMapperOptions.tcl
% run map
% view report
```

## Simplified Synthesis Flow Batch Script

This batch script reproduces the results from [RTL Preparation, on page 18](#), [FDC Constraint Application, on page 18](#), and [Mapper Execution, on page 19](#). The script assumes that the option values were set previously (option values are retained within a design project).

```
% source MyCompilerOptions.tcl
% run compile -file MyRtlFilelist
% source MyGlobalPrepOptionst.tcl
% run pre_map -file MyFdcFile.tcl
% source MyMapperOptions.tcl
% run map
% view report
```

## FPGA Synthesis with Multiple Task Exploration

This section includes examples that show how multiple iterations of a task are used to evaluate performance.

### Multiple Compile Nodes

For this example, assume a complete RTL file list.

```
# Create a compile database state using the default (runtime-
# optimized) compiler. Include the -out option to give the
# database state a unique name ("FST-COMP" in the example):

% source allOptions.tcl
% run compile -filelist MyRtlFilelist -out FST-COMP
% view report

# Create a second compile state using the performance-optimized
# compiler. Include the -out option to give this state a
# unique name ("PERF-OPT" in the example):

% option set synthesis_strategy fast
% run compile -filelist MyRtlFilelist -out PERF-OPT
% view report
```

### Multiple Prepare Tasks

This example uses the compile database states created in [Multiple Compile Nodes](#), on page 20 to generate multiple prepare states.

```
# Using the compile PERM-OPT state, run global prep with GCC
# enabled to create prepare database state PERM_GCC:

% database set PERM-OPT
% option set fix_gated_and_generated_clocks 1
% run pre_map -file MyFdcFile -out PERM_GCC
% view report
```

```
# Do the same as above using database state FST-COMP as input to
# create prepare database state FST_GCC.

% database set FST-COMP
% option set fix_gated_and_generated_clocks 1
% run pre_map -fdc MyFdcFile -out FST_GCC
% view report

# Map both compiler versions. The current database state at this
# point is FST_GCC:

% run map -out MAP_FST_GCC
% database set PERF_GCC
% run map -out MAP_PERF_GCC

# Launch analyst to compare the SRM and SRS files for the two
# compile modes. The first view schematic command brings up the
# SRS file (Technology view) from the map state. The second
# command brings up the SRM file (RTL view) from the pre-map
# state:

% view schematic (from MAP_PERF_GCC state)
% database set PERF_GCC
% view schematic (from PERF_GCC state)

# Now launch analyst to compare the SRM and SRS files for node
# FST_GCC:

% database set FST_GCC
% view schematic
% database set MAP_FST_GCC
% view schematic

# Run estimated timing on the mapped database states:

% report timing
```

# Debug Scripts

The following examples illustrate launching the instrumentor and using IDC files in batch mode:

## Instrumentation Window

This example opens the instrumentation window to create an IDC file.

```
# Use the pre_instrument task to create an rtl_debug node:
% run pre_instrument -filelist MyFilelist

# Open the instrumentation window to create an IDC file:
% edit idc -mode gui

# Instrument and save your design
```

## RTL Instrumentation

This example instruments the RTL in batch mode and assumes an existing IDC file.

```
# Use the pre-instrument task to create rtl_debug node RTLDbg:
% run pre_instrument -filelist MyRtlFilelist -out RTLDbg

# Annotate the IDC (and CDC) information to the SRS netlist.
# Use the RtlDbg database state created from the previous
# command to create the RtlDbgCompile node:
% run compile -idc MyRtlDbg.idc -out RtlDbgCompile
```

## SRS Instrumentation

This example instruments the SRS in batch mode and assumes an existing IDC file with SRS instrumentation directives.

```
# Generate design.srs consistent with the IDC file and using a
# default compile database state:
% run compile -filelist MyRtlFilelist
```

```
# Directly add the SRS instrumentation connections to the SRS.
# Use the RtlDbgCompile state created by the RTL Instrumentation
# example above as input. File MySrsDbg.idc triggers the SRS
# modifications for debug and creates prepare database
# state SrsDbg:

% run pre_map -idc MySrsDbg.idc -out SrsDbg
```

## RTL and SRS Instrumentation

This example instruments the RTL and SRS in batch mode and assumes an existing IDC file with SRS instrumentation directives.

```
# Run the pre-instrument task to create an RtlDbg state:

% run pre_instrument -filelist MyRtlFilelist -out RtlDbg

# Annotate the IDC file (and corresponding tool-generated.cdc)
# information to the SRS. Use the RtlDbg database state from the
# previous command as input to create compile state RtlDbgCompile:

% run compile -idc MyRtlDbg.idc -out RtlDbgCompile

# Add the SRS instrumentation connections to the SRS.
# Use the RtlDbgCompile state from the previous task as input.
#   IDC File MySrsDbg.idc triggers the SRS modifications
#   for debug and creates pre-map state RtlSrsDbg:

% run pre_map -idc MySrsDbg.idc -out RtlSrsDbg
```





## CHAPTER 3 P

# Graphical User Interface

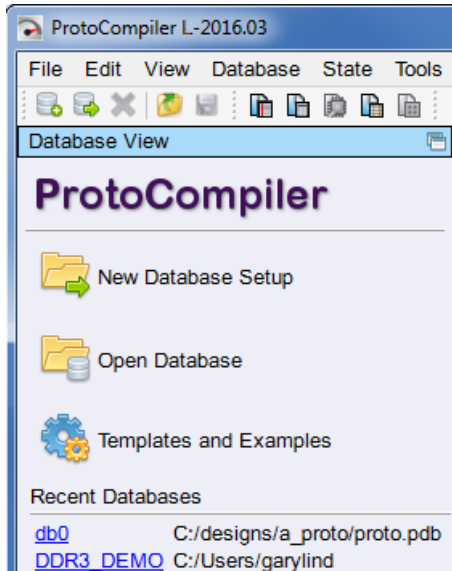
---

The graphical user interface duplicates a number of the Tcl shell commands. Within this interface, dialog boxes and sub-menus are included to offer a complete graphical solution. The remainder of the chapter describes:

- [Graphical Interface](#), on page 26
- [Graphical Window](#), on page 29
- [Menus](#), on page 32
- [Icons](#), on page 85
- [State Tree View Operations](#), on page 88
- [Design Flow View Operations](#), on page 91
- [Secondary Views](#), on page 96
- [Task Dialog Boxes](#), on page 100
- [PCF Graphical Editor](#), on page 116
- [File List Editor](#), on page 128
- [Options Editor](#), on page 134
- [Add Vivado IP](#), on page 136
- [Verdi Debug Setup](#), on page 137

# Graphical Interface

The graphical interface is enabled by default. The figure below shows the initial Database View when the ProtoCompiler is first started.



The following describe the active functions and icons available from the initial Database Design Flow window.

## Dock/Undock Current Window

Undocks the Database Design Flow window. An undocked window is docked by again clicking the icon.

## Close

Closes the Database Design Flow window. The window is reopened by selecting Data View from the View drop-down menu or by pressing Ctrl-D.

## New Database Setup



Opens the Create Database dialog where you enter a database directory location and a name for the database. Clicking OK opens the new database in the Database Design Flow window.

## Open Database

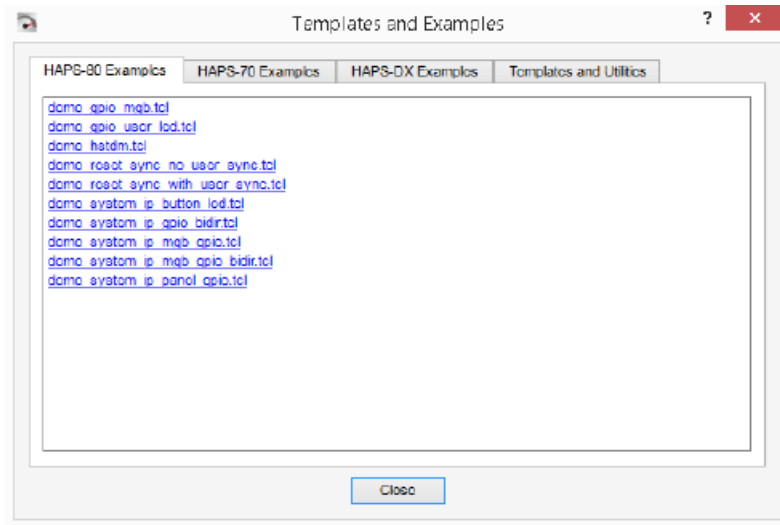


Opens a browser where you can select an existing database directory. Clicking Choose opens the selected database in the Database Design Flow window.

## Templates and Example



Opens the Templates and Examples dialog where you can run prepared examples or Tcl scripts.



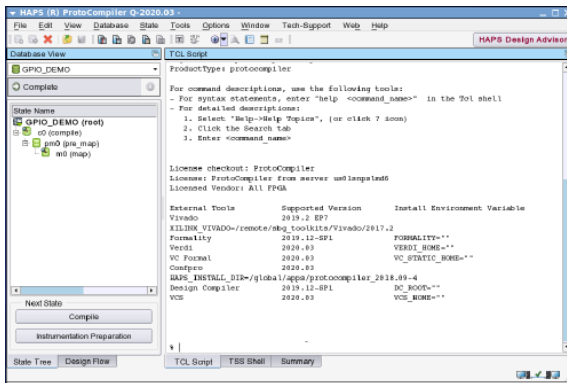
To run an example, click on the example name on one of the tabs, navigate to the directory where the example is to be run, and click Choose. To use a Tcl script, click on the script name to open the Save Template dialog, navigate to the design directory where you want to run the script, and click Save. The script is copied to the specified directory and opened in a text editor window.

## **Recent Databases**

Lists recent database projects. Clicking on a database name opens the database in the Database View window with the existing database states displayed.

# Graphical Window

When a database is specified (either using New Database Setup or by opening an existing database), the Database View window on the left is redisplayed to show the current status of the database. The following figure shows the complete starting window with the updated Database View on the left, the TCL Script window on the right, and a set of drop-down menus and menu icons across the top.



## Database View

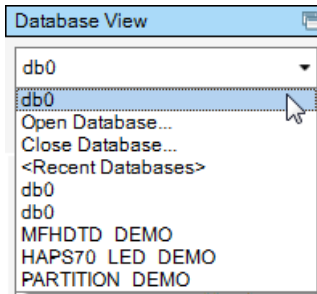
The Database View displays either the current database states within the design or a graphic representation of the design flow according to which tab is selected at the bottom of the view. For descriptions of these tab functions, see:

- [State Tree View Operations](#), on page 88
- [Design Flow View Operations](#), on page 91

These two tabs share the fields at the top of the Database View.

## Database Field Drop-down

Performs basic database functions including opening or closing a database or selecting a database from a list of recent databases. Opening another database prompts before replacing the current database.



## Active Job Flow Field

Reports the currently running task as a result of execution of a run command. The task can be terminated at any time by clicking the adjacent stop button.

## Tcl Script Window

The Tcl Script window is an interactive command shell that implements the Tcl command-line interface. You can type or paste Tcl commands at the prompt. For a list of the available commands, type `help` at the Tcl prompt. For general information about Tcl syntax, select `Help ->TCL`, for detailed information on the ProtoCompiler-supported Tcl commands, see the *ProtoCompiler Command Reference Manual*.

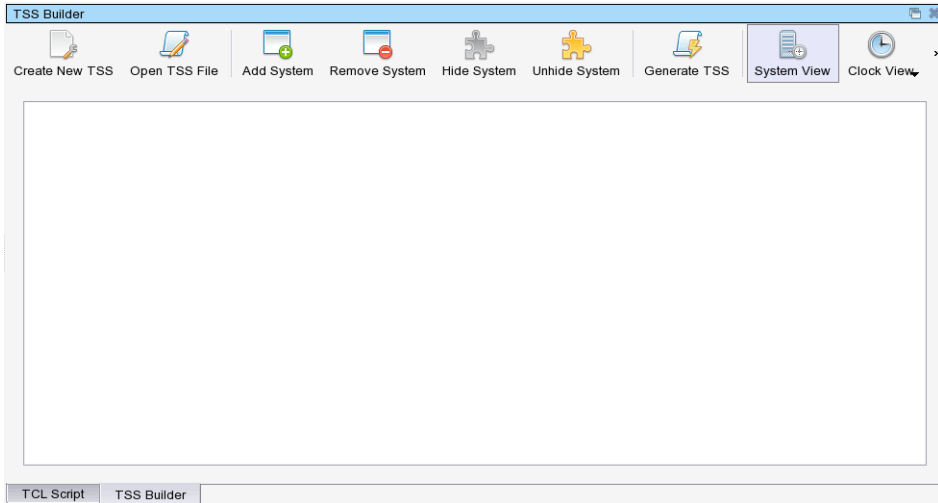
The Tcl script window also displays each command executed in the course of running ProtoCompiler whether it was initiated from a menu, button, or keyboard shortcut. Right-clicking inside the Tcl window displays a popup menu with the basic editing commands.

A secondary TSS Shell is available from the TCL Script window by typing `launch tss` at the Tcl Script prompt. This window is used to enter board-query commands.

## TSS Builder Window

The TSS Builder Window is an interactive window that provides a graphical view of the system configuration in a Target System Specification (TSS) file. Open the window by selecting TSS Builder from the Tools Menu.

TSS Builder enables you to generate new TSS files or modify existing TSS files using graphical user interface. A menu bar with icons and usage information is available at the top of the window.



The System View provides a graphical view of all systems and connections in a TSS file. HAPS Systems are displayed in different colors based on the system information. Connections are color coded based on the cable length. You can select the graphical elements using a mouse. Right-click on various elements for pop-up menus with configuration options specific to the element. You can also enlarge the window to include more graphical elements. The Hide and Unhide option allows you to focus on select systems in the configuration.

For details on TSS Builder, see [TSS Builder, on page 57](#).

For details on Generating and Modifying TSS files using TSS Builder, see [Generating TSS Files Using TSS Builder, on page 228](#) in the User Guide.

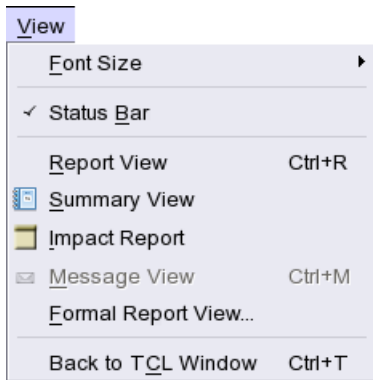
# Menus

A set of drop-down menus are at the top of the graphics window. These menus include:

- [View Menu](#), on page 32
- [Database Menu](#), on page 33
- [State Menu](#), on page 36
- [Tools Menu](#), on page 39 with [Timing Analyst](#), on page 48
- [Options Menu](#), on page 70
- [Tech-Support Menu](#), on page 81
- [Web Menu](#), on page 82
- [Help Menu](#), on page 83

## View Menu

The View menu controls the window display as outlined in the following table.





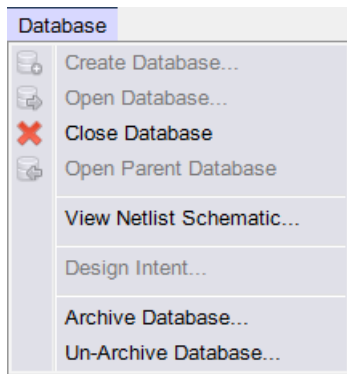
---

Font Size	Increases or decreases the font size in menus from a drop-down menu selection. A Reset Font Size selection returns to the default size.
Status Bar	Displays the status icons in the lower right corner of the window when checked.
Report View	Opens a report view from the active database state. The available reports for each completed state are listed in order. The Report View is also available by clicking the View Reports icon in the menu bar or by entering Ctrl-R../
Summary View	Generates a summary page detailing the results from each completed database state. The Summary View is also available by clicking the View Database Summary icon in the menu bar or by entering Ctrl-A.
Impact Report	Opens the Impact Information Window.
Message View	Opens the Report Messages View which lists the individual messages concerning a database state, see <a href="#">Report Messages View</a> , on page 97. The Report Message View is also available by clicking the View Messages icon or by entering Ctrl-M.
Formality Report View	Opens a dialog to enter the location of an exported formal verification report for viewing.
Back to TCL Window	Reduces any open views to tabs and displays the TCL Script window.

---

## Database Menu

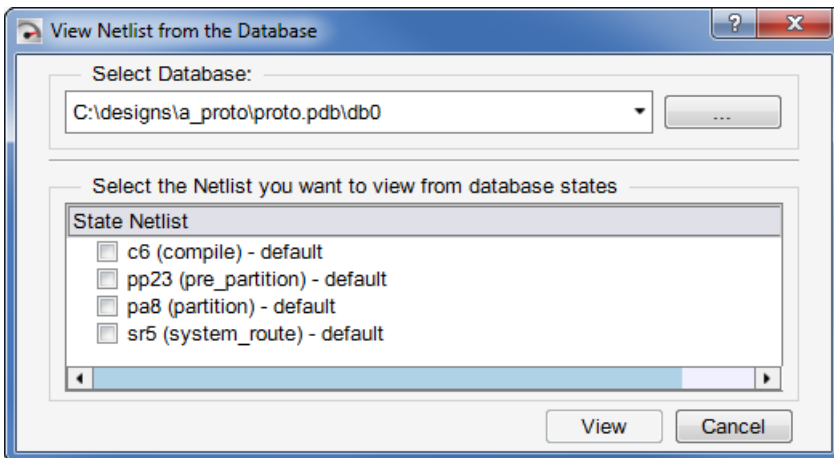
The Database menu performs database operations, opens schematics for viewing, and performs archiving tasks.



Create Database	Opens the Create Database dialog where you enter a database directory location and a name for the database. Clicking OK opens the new database in the Database Design Flow window. The menu selection is enabled when there is no active database and performs the same function as the New Database Setup button or the Create Database icon.
Open Database	Opens a browser where you can select an existing database directory. Clicking Choose opens the selected database in the Database Design Flow window. The menu selection is enabled when there is no active database and performs the same function as the Open Database button or the Open Database icon.
Close Database	Closes the current database and returns to the initial Database Design Flow window. Menu selection prompts to confirm closing and performs the same function as the Close Database icon.
Open Parent Database	Returns to the parent database from the current sub-database.
View Netlist Schematic	Opens the View Schematic from selected Database dialog box to allow selection of a database from the available listed databases. For more information, see <a href="#">View Netlist Schematic from the Database Dialog Box</a> , on page 35.

Design Intent	Opens the Design Intent dialog to allow the optimization strategy to be defined for the ensuing operations. Selection is available only when a database is initially loaded. For more information, see <a href="#">Synthesizing Based on Design Intent</a> , on page 493 in the <i>User Guide</i> .
Archive Database	Opens the first page of the archiving dialog box. See <a href="#">Archiving and Unarchiving Databases</a> , on page 66 in the <i>User Guide</i> .
Un-Archive Database	Opens the un-archiving dialog box. See <a href="#">Archiving and Unarchiving Databases</a> , on page 66 in the <i>User Guide</i> .

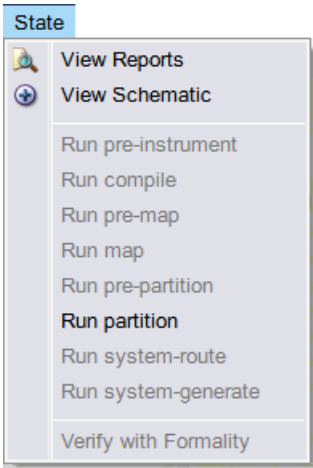
## View Netlist Schematic from the Database Dialog Box



Select Database	Selects the active database. Use the adjacent browse button to locate the desired database; the current database is selected by default.
State Netlist	Lists the schematic netlists available within the selected database.
View button	Enabled when one or more state netlists are checked. Clicking the View button displays the corresponding schematic for the selected netlist state in an HDL analyst window. For information on using the HDL analyst, see <a href="#">Working in the Standard HDL Analyst Schematic</a> , on page 157 in the <i>ProtoCompiler Compiler and Mapper Guide</i> .

# State Menu

The State menu selects the database state to run. Database state selection is determined by both the current state of the design and the design flow.



View Reports	Opens a report view from the active database state. The available reports for each completed state are listed in order. The Report View is also available by clicking the View Reports icon in the menu bar or by entering Ctrl-R. For additional information, see <a href="#">Checking Reports and Log Files, on page 512</a> in the User Guide and <a href="#">Chapter 5, Inputs, Reports and Log Files</a> in the Reference.
View Schematic	Opens a schematic, if available, for the current database state in the adjacent window ( <i>designName</i> tab). The schematic view displayed depends on the current database state; when multiple schematics are available, the adjacent scroll bar selects the schematic for the corresponding database state.
Run pre-instrument	Opens the Run Instrumentation Preparation dialog box. The selection is enabled only from root. For additional information, see <a href="#">Run Instrumentation Preparation Dialog Box , on page 103</a> , <a href="#">Instrumenting the Design Before Compiling, on page 642</a> in the User Guide and <a href="#">run pre_instrument, on page 149</a> in the Command Reference.

---

Run compile	Opens the Run Compile dialog box. The selection is enabled from root or from the pre-instrument state. For additional information, see <a href="#">Run Compile Dialog Box</a> , on page 101, <a href="#">Compiling the Design</a> , on page 77 in the <i>User Guide</i> and <a href="#">run compile</a> , on page 132 in the <i>Command Reference</i> .
Run pre-map	Opens the Run Pre-Map dialog box. The selection is enabled only for the synthesis design flow and only after successful execution of a compile task. For additional information, see <a href="#">Run Pre-Map Dialog Box</a> , on page 105, <a href="#">Running Pre-Map</a> , on page 474 in the <i>User Guide</i> and <a href="#">run pre_map</a> , on page 153 in the <i>Command Reference</i> .
Run map	Opens the Run Map dialog box. The selection is enabled only for the synthesis design flow and only after successful execution of a pre-map task. For additional information, see <a href="#">Run Map Dialog Box</a> , on page 107, <a href="#">Mapping the Design</a> , on page 490 in the <i>User Guide</i> and <a href="#">run map</a> , on page 142 in the <i>Command Reference</i> .
Run pre-partition	Opens the Run Pre-Partition dialog box. The selection is enabled only for the partition design flow and only after successful execution of a compile task. For additional information, see <a href="#">Run Pre-Partition Dialog Box</a> , on page 108, <a href="#">Using run_pre_partition to Define Partitions</a> , on page 324 in the <i>User Guide</i> , and <a href="#">run_pre_partition</a> , on page 155 in the <i>Command Reference</i> .
Run partition	Opens the Run Partition dialog box. The selection is enabled only for the partition design flow and only after successful execution of a pre-partition task. For additional information, see <a href="#">Run Partition Dialog Box</a> , on page 109, <a href="#">Partitioning the Logic</a> , on page 328 in the <i>User Guide</i> , and <a href="#">run partition</a> , on page 144 in the <i>Command Reference</i> .

---

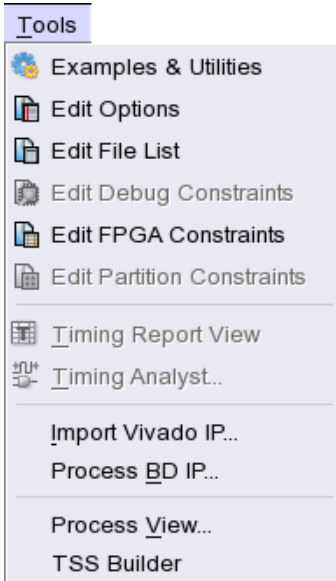
---

Run system-route	Opens the Run System Route dialog box. The selection is enabled only for the partition design flow and only after successful execution of a partition task. For additional information, see <a href="#">Run System Route Dialog Box</a> , on page 111, <a href="#">Running System Route for the Top Level</a> , on page 412 in the <i>User Guide</i> and <a href="#">run system_route</a> , on page 160 in the <i>Command Reference</i> .
Run system-generate	Opens the Run System Generate dialog box. The selection is enabled only for the partition design flow and only after successful execution of a system-route task. For additional information, see <a href="#">Run System Generate Dialog Box</a> , on page 113, <a href="#">Generating FPGAs</a> , on page 416 in the <i>User Guide</i> , and <a href="#">run system_generate</a> , on page 157 in the <i>Command Reference</i> .
Verify with Formality	Starts the Formality tool. For additional information, see <a href="#">Verifying Single-FPGA Designs</a> , on page 945 in the <i>User Guide</i> .

---

## Tools Menu


The Tools menu accesses tools for setting up and controlling the database flow through a set of editable input files.



Templates & Examples	Opens the Templates and Examples dialog where you can run prepared examples or Tcl scripts, see <a href="#">Templates and Example</a> , on page 27.
Edit Options	Opens the Options Editor dialog to allow you to change the default option settings. For more information on the PCF editor, see <a href="#">Options Editor</a> , on page 134.
Edit File List	Prompts for a new or existing file-list file and then opens the file list editor. For more information on the PCF editor, see <a href="#">File List Editor</a> , on page 128.
Edit Debug Constraints	Opens the selected IDC file in either the instrumentor GUI or in a text editor.
Edit FPGA Constraints	Prompts for an FDC constraint file and then opens the file in the constraints editor GUI. For more information on the constraints editor, see <a href="#">Using the Constraints Editor</a> , on page 16 in the <i>ProtoCompiler Compiler and Mapper Guide</i> .

Edit Partition Constraints	Prompts for a PCF file and then opens the PCF editor GUI. For more information on the PCF editor, see <a href="#">PCF Graphical Editor</a> , on page 116.
Timing Report View	The Timing Report View displays timing reports, lets you view and query critical timing paths, and correlate your timing results, see <a href="#">Timing Report View</a> , on page 40.
Timing Analyst	The Timing Analyst or timing analyzer lets you generate custom reports for paths, see <a href="#">Timing Analyst</a> , on page 48.
Import Vivado IP	Prompts for a Vivado IP file or directory for import. For more information on adding Vivado files, see <a href="#">Add Vivado IP</a> , on page 136.
Process BD IP	Prompts for a Vivado block design (BD) file to automatically process and convert it to a DCP file. You can then use the DCP file to import the IP. See <a href="#">Importing IP from a Block Design File</a> , on page 356 in the <i>ProtoCompiler Compiler and Mapper Guide</i> .
Process View	Opens the Process View window that shows all processes that are running and that have been run.
TSS Builder	Opens the TSS Builder Window. The TSS Builder enables you to generate or modify Target System Specification (TSS) files for prototyping projects that involves multiple HAPS systems and daughter boards. The TSS Builder Window displays a graphical representation of the Systems, Daughter Cards, Connections, and Clocks defined in the TSS files. See <a href="#">TSS Builder</a> , on page 57.

## Timing Report View

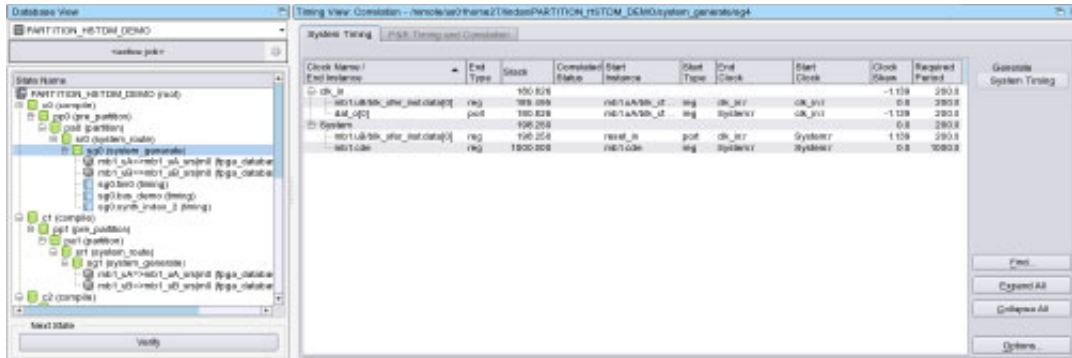
Use the Tools->Timing Report View command or its associated icon (  ) to display the Timing Report View tool. The Timing Report View displays timing reports, lets you view and query critical timing paths, and correlate your timing results. You can compare synthesis timing results with P&R Static Timing Analysis (STA) results and determine if paths for timing end points, start points, and requested periods match.

For more information, see [Viewing and Correlating Results with the Timing Report View](#), on page 535 in the *ProtoCompiler User Guide*.



## System Timing Tab

System Timing results are displayed from System Generate state.



The following table describes the options you can set on the System Timing tab.

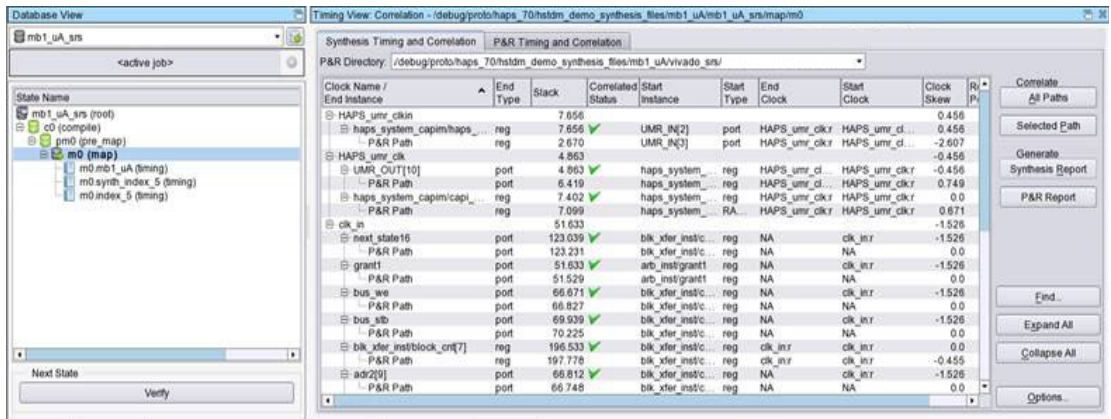
Field	Description
<b>System Timing Options</b>	
Generate	
• System Timing Button	Generate the system timing report for the selected path.
Find	Locate clock and instance names in this view.
Expand All	Expand all paths for the specified clock or instance.
Collapse All	Collapse all paths for the specified clock or instance.
Options	Specify the number of P&R paths per end point to include in the synthesis timing report. The default is 1. For details, see <a href="#">Synthesis Timing Options</a> , on page 44.
<b>System Timing Report</b>	
Clock Name / End Instance	Clock name or name of end point instance

Field	Description
End Type	Type of end point instance from the P&R tool that can be specified as <ul style="list-style-type: none"><li>• reg</li><li>• port</li></ul>
Slack	Slack value
Correlated Status	Status of timing report paths: <ul style="list-style-type: none"><li>• Green Check Mark- Paths passed correlation</li><li>• Red X Mark- Paths failed correlation - Float over the cell for a tool tip describing the error.</li><li>• Gray - Correlation not run for paths</li></ul>
Start Instance	Name of start point instance
Start Type	Type of start point instance from the P&R tool that can be specified as <ul style="list-style-type: none"><li>• reg</li><li>• port</li></ul>
End Clock	End rising/falling clock
Start Clock	Start rising/falling clock
Clock Skew	Clock skew value
Required Period	Requested period

## Synthesis Timing and Correlation Tab

The Timing Report View runs Vivado place and route. You can access place-and-route timing paths that can be compared and correlated with their corresponding synthesis paths. When the synthesis path correlates to a place-and-route path, the P&R path is displayed below the synthesis path. Synthesis Timing results are displayed from the Map state. Synthesis timing is considered the golden timing when correlating to P&R when in the Synthesis Timing and Correlation tab.

The Synthesis Timing and Correlation and P&R Timing and Correlation tabs are mutually exclusive. The panel in front is active and uses the selected paths to run the options for the synthesis timing correlation report or the P&R timing correlation report.



The Synthesis Timing and Correlation and P&R Timing and Correlation tabs are mutually exclusive. The panel in front is active and uses the selected paths to run the options for the synthesis timing correlation report or the P&R timing correlation report.

The following table describes the options you can set on the Synthesis Timing and Correlation tab.

Field	Description
<b>Synthesis Timing Options</b>	
P&R Directory	Select the P&R directory for an implementation or other project.
Correlate <ul style="list-style-type: none"> <li>All Paths Button</li> <li>Selected Path Button</li> </ul>	Run the timing correlation report for all paths or only selected paths.
Generate <ul style="list-style-type: none"> <li>Synthesis Report Button</li> <li>P&amp;R Report Button</li> </ul>	<ul style="list-style-type: none"> <li>Generate the synthesis timing report for the selected path.</li> <li>Display the P&amp;R report file (<i>twr</i>) for the selected path.</li> </ul>
Find	Locate clock and instance names in this view.
Expand All	Expand all paths for the specified clock or instance.

Field	Description
Collapse All	Collapse all paths for the specified clock or instance.
Options	Specify the number of P&R paths per end point to include in the synthesis timing report. The default is 1. For details, see <a href="#">Synthesis Timing Options</a> , on page 44.
<b>Synthesis Timing Report</b>	
Clock Name / End Instance	Clock name or name of end point instance
End Type	Type of end point instance from the P&R tool that can be specified as <ul style="list-style-type: none"> <li>• reg</li> <li>• port</li> </ul>
Slack	Slack value
Correlated Status	Status of timing report paths: <ul style="list-style-type: none"> <li>• Green Check Mark- Paths passed correlation</li> <li>• Red X Mark- Paths failed correlation - Float over the cell for a tool tip describing the error.</li> <li>• Gray - Correlation not run for paths</li> </ul>
Start Instance	Name of start point instance
Start Type	Type of start point instance from the P&R tool that can be specified as <ul style="list-style-type: none"> <li>• reg</li> <li>• port</li> </ul>
End Clock	End rising/falling clock
Start Clock	Start rising/falling clock
Clock Skew	Clock skew value
Required Period	Requested period

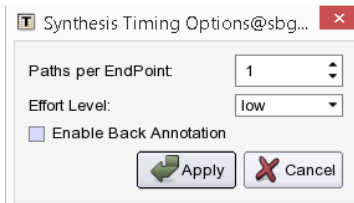
## Synthesis Timing Options

The Synthesis Timing Options dialog box lets you specify the number of paths per end point to include in the synthesis timing report.

For the Synthesis Timing and Correlation that correlates to place and route results, you can specify the following effort levels:

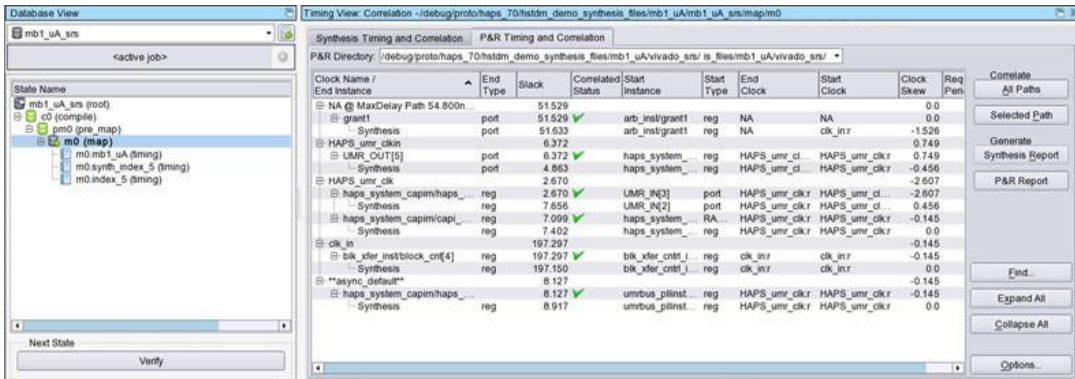
- **low** - Runs a default timing report for Vivado (no -from/-to filtering). From this report a first-pass correlation is run. For the paths that did not correlate, switch to effort medium and rerun for All Paths, or select individual paths and run Selected Path.
- **medium** - Performs a low effort run, but continues to run the second-pass correlation until there is no further convergence of results. This occurs when the number of correlation failures does not change from the previous run.
- **high** - Performs a medium effort level run. However, whenever convergence stops, it then continues running correlation on the remaining individual paths until all have been attempted. Be aware that effort level high can incur large runtimes, because Vivado is called repeatedly.

Check the Enable Back Annotation box to enable using back-annotated timing data. Note that if you have not done the back annotation process, report timing will produce an error.



## P&R Timing Correlation Tab

The P&R Timing Correlation tab allows you to set options for timing.



The Synthesis Timing and Correlation and P&R Timing and Correlation tabs are mutually exclusive. The panel in front is active and uses the selected paths to run the options for the synthesis timing report or the P&R timing correlation report. P&R Timing results are displayed and they are considered golden when correlating to Synthesis when in P&R Timing and Correlation tab.

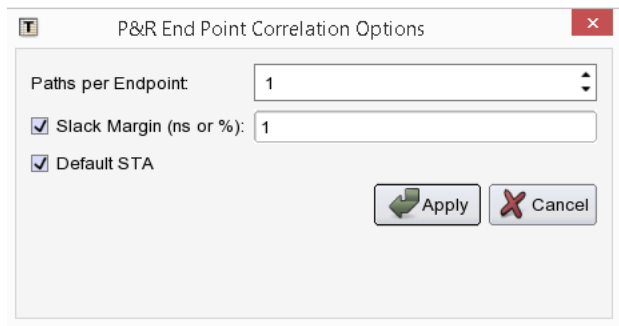
The following table describes the options you can set on the P&R Timing and Correlation tab.

Field	Description
<b>P&amp;R Timing Options</b>	
P&R Directory	Select the P&R directory for an implementation or other project.
Correlate	Run the timing correlation report for all paths or only selected paths.
<ul style="list-style-type: none"> <li>• All Paths Button</li> <li>• Selected Path Button</li> </ul>	
Generate	
<ul style="list-style-type: none"> <li>• Synthesis Report Button</li> <li>• P&amp;R Report Button</li> </ul>	<ul style="list-style-type: none"> <li>• Generate the synthesis timing report for the selected path.</li> <li>• Display the P&amp;R report file for the selected path.</li> </ul>
Find	Locate clock and instance names in this view.
Expand All	Expand all paths for the specified clock or instance.
Collapse All	Collapse all paths for the specified clock or instance.

Options	Set P&R timing correlation options. For more information, see <a href="#">P&amp;R Timing and Correlation Options</a> , on page 48.
<b>P&amp;R Timing Report</b>	
Clock Name/ End Instance	Paths listed by end point for each UCF Constraint.
End Type	Type of end point instance from the P&R tool that can be specified as: <ul style="list-style-type: none"> <li>• reg</li> <li>• port</li> </ul>
Slack	Slack value
Correlated Status	Status of timing report paths: <ul style="list-style-type: none"> <li>• Green Check Mark- Paths passed correlation</li> <li>• Red X Mark- Paths failed correlation - Float over the cell for a tool tip describing the error.</li> <li>• Gray - Correlation not run for paths</li> </ul>
Start Instance	Name of the start point instance
Start Type	Type of start point instance from the P&R tool that can be specified as: <ul style="list-style-type: none"> <li>• reg</li> <li>• port</li> </ul>
End Clock	End rising/falling clock
Start Clock	Start rising/falling clock
Clock Skew	Clock skew value
Required Period	Requested period

## P&R Timing and Correlation Options


The P&R End Point Correlation Options dialog box lets you specify the options to narrow the scope of paths included in the timing correlation report.



Enable any of the options in the following table:

Options	Description
Paths per Endpoint	Specify the number of P&R paths per end point to include in the synthesis timing report. The default is 1.
Slack Margin	This option directs the timing comparison routines to compare the reported slack for a P&R path to the same path in synthesis, if found. When a P&R path successfully correlates to a synthesis path, this option further checks that reported slacks are within the given range (in nanoseconds) of each other. When enabled, slacks that are within the given range show up as <b>green</b> and slacks out of range show up as <b>red</b> . Use the tool tip to fly over red slacks with the cursor to see a report of the slack found in synthesis.
Default STA	Use the default timing report from P&R to populate the End Point/Start Point Table. Use the option early on to verify the P&R critical paths in a design. This is the default.

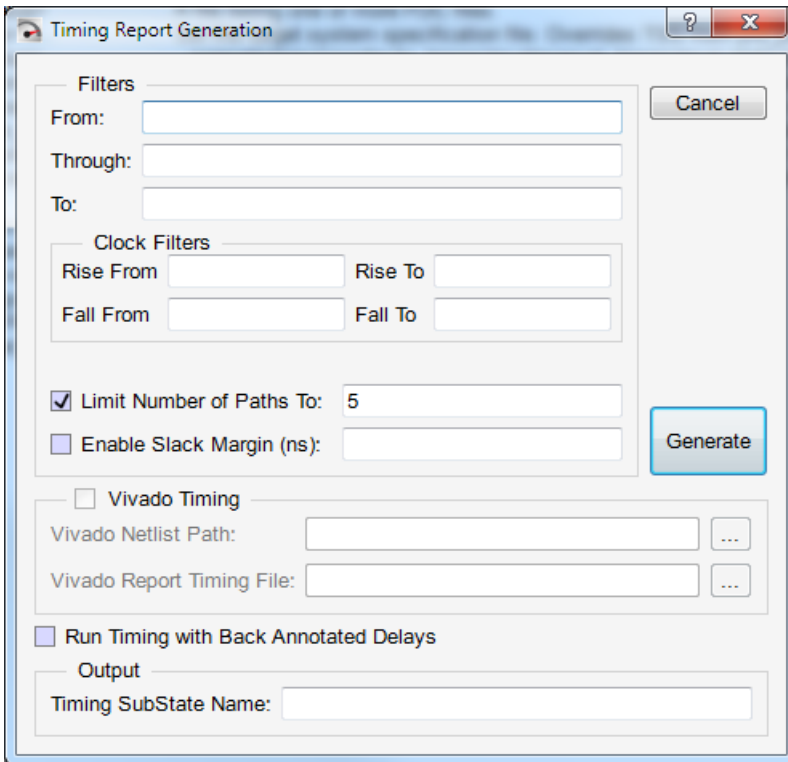
## Timing Analyst

The Timing Analyst or timing analyzer lets you generate custom reports for paths. You can access the timing analyst by selecting Tools->Timing Analyst or by clicking the icon (  ). See the following for more details:



- [Timing Report Generation Dialog Box](#), on page 49
- [Timing Analyst Through Points](#), on page 51
- [Path Filter Combinations for the Timing Analyst](#), on page 52
- [Timing Analyst Examples with Wildcards](#), on page 55
- [Generating Custom Timing Reports with the Timing Analyst](#), on page 541 in the *User Guide*

## Timing Report Generation Dialog Box



The image shows a screenshot of the 'Timing Report Generation' dialog box. It has a title bar with a question mark and a close button. The dialog is divided into several sections:

- Filters:** Contains three text input fields labeled 'From:', 'Through:', and 'To:'.
- Clock Filters:** Contains four text input fields arranged in two rows: 'Rise From' and 'Rise To' in the first row, and 'Fall From' and 'Fall To' in the second row.
- Limit Number of Paths To:** A checkbox is checked, followed by a text input field containing the value '5'.
- Enable Slack Margin (ns):** An unchecked checkbox followed by a text input field.
- Generate:** A blue button located to the right of the 'Enable Slack Margin' section.
- Cancel:** A button located to the right of the 'Filters' section.
- Vivado Timing:** An unchecked checkbox.
- Vivado Netlist Path:** A text input field followed by a browse button (three dots).
- Vivado Report Timing File:** A text input field followed by a browse button (three dots).
- Run Timing with Back Annotated Delays:** An unchecked checkbox.
- Output:** A section containing a text input field labeled 'Timing SubState Name:'.

The following table provides brief descriptions of the parameters for generating a custom timing report. See [Generating Custom Timing Reports with the Timing Analyst](#), on page 541 in the *User Guide* for information on generating timing reports with the Timing Analyst.

Option	Description
From or To	<p>Specifies the starting (From) or ending (To) point of the path for one or more objects. It must be a timing start point (From) or end (To) point for each object. Use this option in combination with the others in the Filters section of the dialog box. See <a href="#">Path Filter Combinations for the Timing Analyst</a> , on page 52 for examples of using filter combinations.</p> <p>You can use the get command to specify names. See <a href="#">Specifying From, To, and Through Points for Custom Timing Reports</a>, on page 544 in the <i>User Guide</i> for information on object names.</p>
Through	<p>Reports all paths through the specified point or list of objects. Use this option in combination with the others in the Filters section of the dialog box. See the following for additional information:</p> <ul style="list-style-type: none"> <li>• <a href="#">Timing Analyst Through Points</a>, on page 51</li> <li>• <a href="#">Path Filter Combinations for the Timing Analyst</a>, on page 52.</li> </ul>
Clock Filters: Rise From/To Fall From/To	<p>Specifies the rise from/to and fall from/to points for clock objects or clock aliases. See <a href="#">Specifying From, To, and Through Points for Custom Timing Reports</a>, on page 544 in the <i>User Guide</i> for information on object names.</p>
Limit Number of Paths to	<p>Specifies the maximum number of paths to report. The default is 5. Use this option in combination with the others in the Filters section of the dialog box. See <a href="#">Number and Slack Path Filters</a> , on page 52 for additional information.</p>
Enable Slack Margin (ns)	<p>Limits the report to paths within the specified distance of the critical path. Use this option in combination with the others in the Filters section of the dialog box. See <a href="#">Number and Slack Path Filters</a> , on page 52 for additional information.</p>
Vivado Timing	<p>When enabled, uses the Vivado timing engine to run timing analysis instead of the Timing Analyst.</p>
Vivado Netlist Path	<p>Specifies the path for the Vivado netlist. This is required to run Vivado timing analysis</p>
Vivado Report Timing File	<p>Specifies a location for the timing report generated by Vivado timing analysis. Specified paths are relative to the current working directory. If no path is specified, the results are printed in the Tcl window for the prototyping tool.</p>

Option	Description
Run Timing with Back Annotated Delays	When selected, enables running timing analysis with delay values back annotated from Place and Route. See <a href="#">Importing Place and Route Results for Backannotation</a> , on page 584 in the <i>User Guide</i> .
Timing SubState name	Specifies a name for the substate generated by timing analysis.
Generate	Generates the specified timing report.

## Timing Analyst Through Points

You can specify through points for nets (n:), hierarchical ports (t:), or instantiated cell pins (t:).

**OR list** Enter the points as a space-separated list. The points are treated as an OR list and paths are reported if they crosses any of the points in the list. For example, when you type the following, the tool reports paths that pass through points b or c:

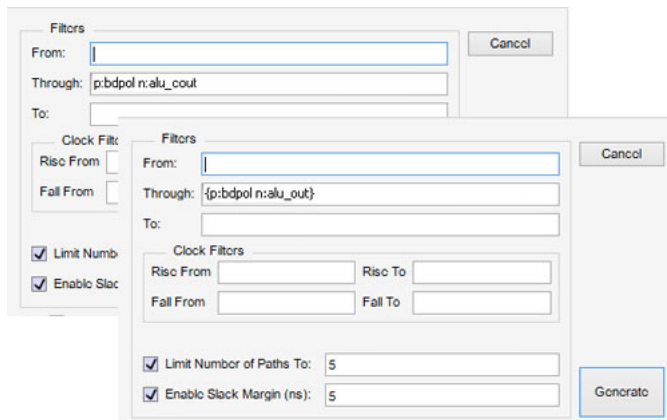
```
{n:b n:c}
```

See [Through Points: OR List Filter for Timing Analyst](#) , on page 51.

See [Specifying From, To, and Through Points for Custom Timing Reports](#), on page 544 in the *User Guide* for more information about specifying through points.

## Through Points: OR List Filter for Timing Analyst

This example reports the five worst paths through port bdpol or net aluout. You can enter the through points as a space-separated list (enclosing the list in braces is optional.)



## Path Filter Combinations for the Timing Analyst

This section describes how to use a combination of path filters to specify what you need and how to specify start and end points for path filtering.

### Number and Slack Path Filters

The Limit Number of Paths To option specifies the maximum number of paths to report and the Enable Slack Margin option limits the report to output only paths that have a slack value that is within the specified value. When you use these two options together, the tighter constraint applies, so that the actual number of paths reported is the minimum of the option with the smallest value. For example, if you set the number of paths to report to 10 and the slack margin for 1 ns, if the design has only five paths within 1 ns of critical, then only five paths are reported (not the 10 worst paths). But if, for example, the design has 15 paths within a 1 ns of critical, only the first 10 are reported.

### From/To/Through Filters

You can specify the from/to points for a path. You can also specify just a from point or just a to point. The from and to points are one or more hierarchical names that specify a port, register, pin on a register, or clock as object (clock alias). Ports and instances can have the same names, so prefix the name with p: for top-level port, i: for instance, or t: for hierarchical port or instance pin. However, the c: prefix for clocks is required for paths to be reported.

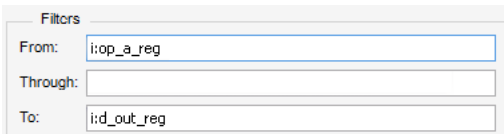
The timing analyst searches for the from/to objects in the following order: clock, port, bit port, cell (instance), net, and pin. Always use the prefix qualifier to ensure that all expected paths are reported. Remember that the timing analyst stops at the first occurrence of an object match. For buses, all possible paths from the specified start to end points are considered.

You can specify through points for nets, cell pins, or hierarchical ports.

You can simply type in from/to or through points. You can also cut-and-paste or drag-and-drop valid objects from the compiled or mapped views into the appropriate fields on the Timing Report Generation dialog box. Timing analysis requires that constraints use the Tech View name space. Therefore, it is recommended that you cut-and-paste or drag-and-drop objects from the mapped view rather than the compiled view.

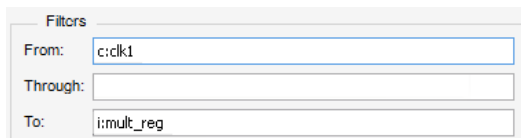
The following examples show how to specify start, end, or through point combinations for path filtering.

### From/To Filter Points: Single Register to Single Register



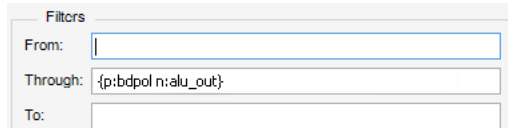
The screenshot shows a dialog box titled "Filters" with three input fields. The "From:" field contains the text "i:op\_a\_reg". The "Through:" field is empty. The "To:" field contains the text "i:d\_out\_reg".

### From/To Filter Points: Clock Object to Single Register



The screenshot shows a dialog box titled "Filters" with three input fields. The "From:" field contains the text "c:clk1". The "Through:" field is empty. The "To:" field contains the text "i:mult\_reg".

### From/To Filter Points: Single Bit of a Bus to Single Register



The screenshot shows a dialog box titled "Filters" with three input fields. The "From:" field is empty. The "Through:" field contains the text "{p:bdpol n:alu\_out}". The "To:" field is empty.

## From/To Filter Points: Single Bit of a Bus to Single Bit of a Bus

The screenshot shows a dialog box titled "Filters". It contains three input fields: "From:" with the value "p:op\_reg[4]", "Through:" which is empty, and "To:" with the value "p:d\_out\_reg[6]".

## From/To Filter Points: Multiple Bits of a Bus to Multiple Bits of a Bus

The screenshot shows a dialog box titled "Filters". It contains three input fields: "From:" with the value "p:op\_a\_reg[7:0]", "Through:" which is empty, and "To:" with the value "p:d\_out\_reg[15:0]".

## From/To Filter Points: With Hierarchy

This example reports the five worst paths for the net foo:

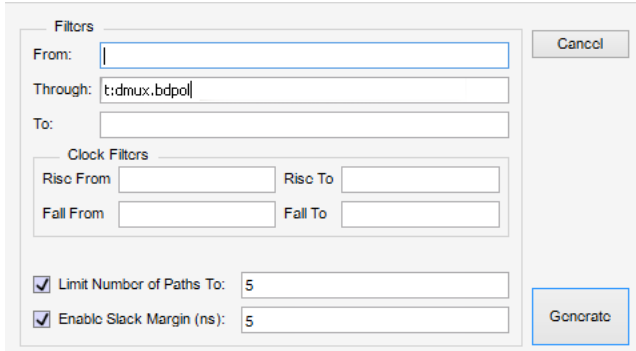
The screenshot shows a dialog box titled "Filters". It contains three input fields: "From:" with the value "i:nac\_core.rxu\_fifo.reg", "Through:" which is empty, and "To:" with the value "i:nac\_core.rxu\_channel.reg". Below these fields is a section titled "Clock Filters" with four input fields: "Rise From", "Rise To", "Fall From", and "Fall To", all of which are empty. At the bottom, there are two checkboxes: "Limit Number of Paths To:" which is checked and has the value "5" in its adjacent input field, and "Enable Slack Margin (ns):" which is unchecked. To the right of the dialog box are two buttons: "Cancel" and "Generate".

## Through Filter Points: Through a Net

The screenshot shows a dialog box titled "Filters". It contains three input fields: "From:" which is empty, "Through:" with the value "n:nac\_core.foo", and "To:" which is empty.

## Through Filter Points: Through a Hierarchical Port

This example reports the five worst paths for the hierarchical port bdpol:



The screenshot shows a 'Filters' dialog box with the following fields and controls:

- Filters** section:
  - From:** An empty text box.
  - Through:** A text box containing the text 't:dmux,bdpol'.
  - To:** An empty text box.
- Clock Filters** section:
  - Rise From:** An empty text box.
  - Rise To:** An empty text box.
  - Fall From:** An empty text box.
  - Fall To:** An empty text box.
- Limit Number of Paths To:** A checkbox that is checked, followed by a text box containing the value '5'.
- Enable Slack Margin (ns):** A checkbox that is checked, followed by a text box containing the value '5'.
- Buttons:** A 'Cancel' button at the top right and a 'Generate' button at the bottom right.

## Timing Analyst Examples with Wildcards

You can use the question mark (?) or asterisk (\*) wildcard characters for object searching and name substitution. These characters work the same way in the synthesis tool environment as in the Linux environment.

### The ? Wildcard

The ? matches single characters. If a design has buses `op_a[7:0]`, `op_b[7:0]`, and `op_c[7:0]`, and you want to filter the paths starting at each of these buses, specify the start points as `op_?[7:0]`. See [Timing Analyst Example: ? Wildcard in the Name](#), on page 56 for another example.

### The \* Wildcard

The \* matches a string of characters. In a design with buses `op_a2[7:0]`, `op_b2[7:0]`, and `op_c2[7:0]`, where you want to filter the paths starting at each of these objects, specify the start points as `op_*`. The report shows all paths beginning at each of these buses and for all of the bits of each bus. See [Timing Analyst Example: \\* Wildcard in the Name \(With Hierarchy\)](#), on page 56 and [Timing Analyst Example: \\* Wildcard in the Bus Index](#), on page 56 for more examples.

### Timing Analyst Example: ? Wildcard in the Name

The ? is not supported in bus indices.

Filters

From:

Through:

To:

### Timing Analyst Example: \* Wildcard in the Name (With Hierarchy)

This example reports the five worst paths, starting at block rxu\_fifo and ending at block rxu\_channel within module nac\_core. Each register in the design has the characters reg in the name.

Filters

From:

Through:

To:

Clock Filters

Rise From  Rise To

Fall From  Fall To

☒ Limit Number of Paths To:

☒ Enable Slack Margin (ns):

Cancel

Generate

### Timing Analyst Example: \* Wildcard in the Bus Index

This example reports the five worst paths, starting at op\_b, and ending at d\_out, taking into account all bits on these buses.

Filters

From:

Through:

To:

Clock Filters

Rise From  Rise To

Fall From  Fall To

☒ Limit Number of Paths To:

☒ Enable Slack Margin (ns):

Cancel

Generate



## TSS Builder

Creating a TSS file manually with several system connections, clocks, and I/O connections may be a tedious task when there are a number of HAPS systems in a prototyping project.

The TSS Builder enables you to view a graphical representation of the systems, daughter boards, connections, and clocks defined in a TSS file. Using the TSS Builder, you can modify TSS files graphically, or generate new TSS files. TSS Builder helps you avoid wrong connections. Required board description files can also be generated automatically.

For details on generating or modifying a TSS File using TSS Builder, see [Generating TSS Files Using TSS Builder, on page 228](#) in the *User Guide*.

TSS Builder supports:

- Reading entire system and displaying the configuration graphically.
- Configuring inter-board connections with clocks and CDE chaining.
- Creating TSS file within the TSS Builder window manually.
- Connecting several HAPS systems and daughter boards.
- Checking cable connections.

TSS Builder supports several HAPS Systems, Daughter Boards, and HapsTrak 3 cables. Following sections provides details on supported HAPS systems, daughter boards, cables, and limitations.

- [Supported Systems, on page 57](#)
- [Supported Daughter Boards, on page 58](#)
- [Supported Cables, on page 58](#)
- [Limitations, on page 59](#)

### Supported Systems

The TSS Builder supports the following HAPS Systems:

HAPS-70 S12	HAPS-80 S26	HAPS-80D Single
HAPS-70 S24	HAPS-80 S52	HAPS-80D Dual
HAPS-70 S48	HAPS-80 S104	






## Supported Daughter Boards

The TSS Builder supports the following Daughter Boards:

- BREAKOUT\_HT3
- CON\_HT3
- DDR3\_SODIMM2R\_HT3
- DDR3\_SODIMM\_HT3
- DDR4\_HT3
- DTD\_MEM
- ECDB
- EXT\_CABLE40\_HT3
- FMC\_ADAPTER\_HT3
- FMCL\_CABLE\_HT3
- GPIO\_HT3
- HTII\_ADAPTER2\_HT3
- HTII\_ADAPTER\_HT3
- HTII\_LS\_ADAPTER\_HT3
- JUNO\_ADAPTER\_HT3
- LAB\_HT3
- LAI\_HT3\_SE
- LPDDR3\_HT3
- SRAM\_HT3
- UMRBUS\_IF

## Supported Cables

TSS Builder supports HapsTrak 3 cables with various lengths. Cables are represented in different colors based on their length.

CON_CABLE_25_HT3	HapsTrak 3, 25 meter connection cable.	
CON_CABLE_50_HT3	HapsTrak 3, 50 meter connection cable.	
CON_CABLE_100_HT3	HapsTrak 3, 100 meter connection cable.	
CON_CABLE_150_HT3	HapsTrak 3, 150 meter connection cable.	
CON_CABLE_200_HT3	HapsTrak 3, 200 meter connection cable.	

## Limitations

- Auto cabling is not supported.
- If you are modifying a TSS file that was not created using a TSS Builder:
  - Option **-auto** for board\_system\_create -interconnect -manual cable settings is supported. But, lines with such configuration details are preserved and added to the generated TSS file.
  - A warning is displayed if there are references to daughter boards that are not supported by TSS Builder. But, the references are omitted from the TSS file generated using the TSS Builder. You may have to add these references manually.
- On a HAPS-80 system, you cannot configure connectors A1, A2, and A3. TSS Builder displays these connectors in white.
- If you are creating TSS file for HAPS-80D:
  - Only two HAPS80D systems can be added to the TSS.
  - You cannot add HAPS-80D with any other system or the UMRBus interface.

- You cannot configure connectors A1, A2, A3, and A12. TSS Builder displays these connectors in white.

## Menu and Icons

The TSS Builder Window displays the following Icons:

### Create New TSS

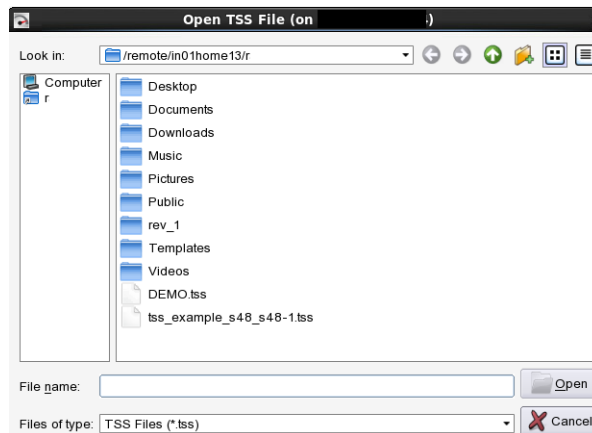


Provides a blank TSS window to start creating a new system configuration. You can Open a TSS file and modify it, or use Add System to create a completely new TSS file.

### Open TSS File



Opens the folder to select an existing TSS File.

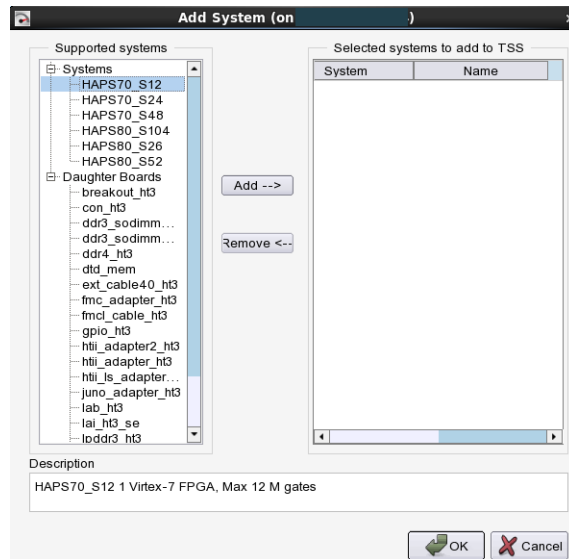


You can view a graphical rendition of the selected TSS file and further modify the file using the TSS builder.

### Add System



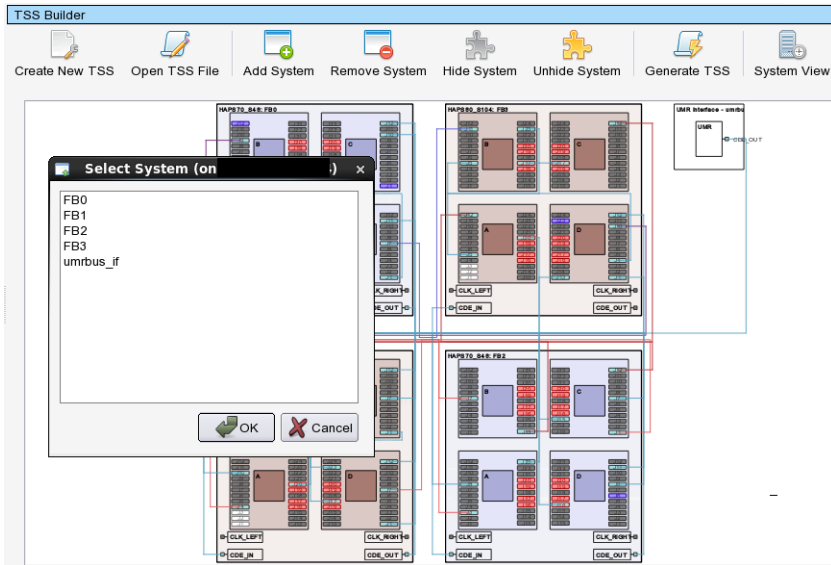
Opens an Add System dialog box to add supported HAPS systems and daughter boards to the system configuration.



## Remove System



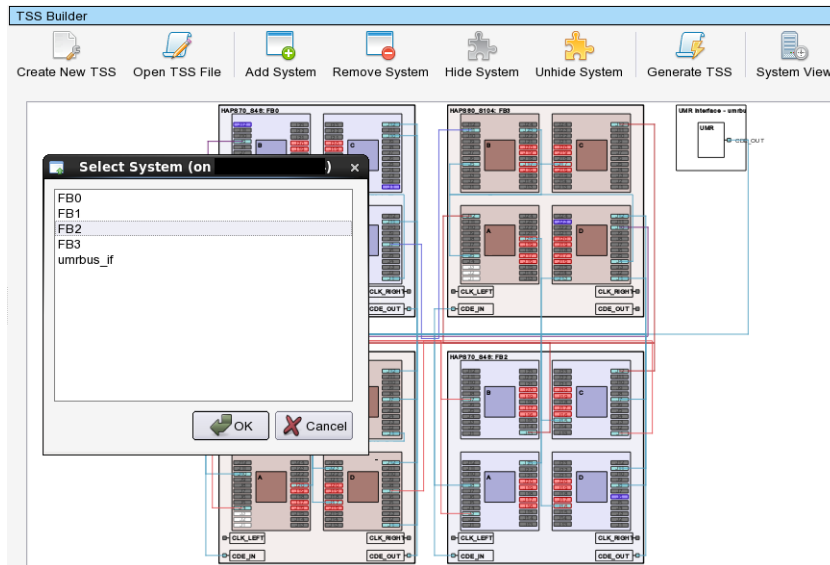
Opens a Select System dialog box with a list of HAPS systems and daughter boards added to the system configuration.



## Hide System



Opens the Select System dialog box with a list of all systems in the configuration. Select the systems from the list to hide them in the GUI. This also hides all connections to and from the selected system.

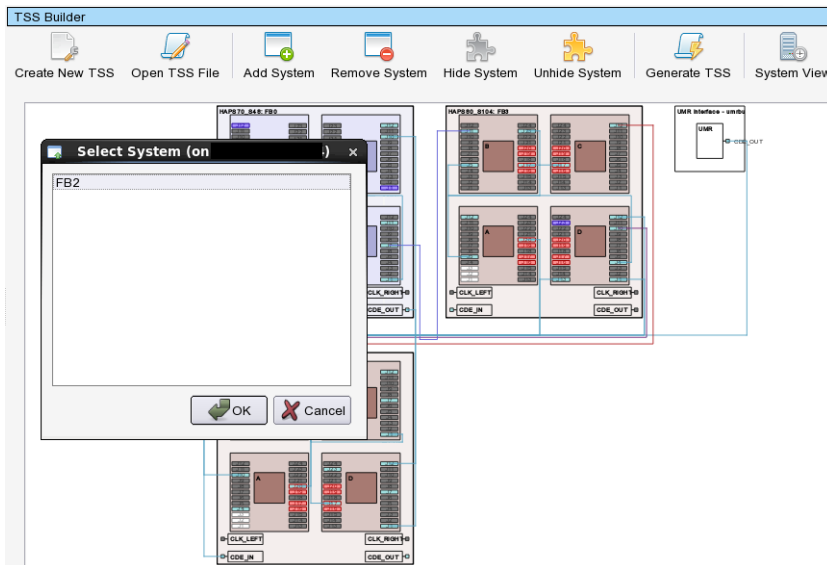


This helps you focus on specific systems and connections when you have a large number of systems in the configuration.

## Unhide System



Opens the Select System dialog box with a list of hidden systems in the configuration. Select the systems you want to view in the GUI.



## Generate TSS



Generates a TSS file from the system configuration done graphically in the TSS Builder.

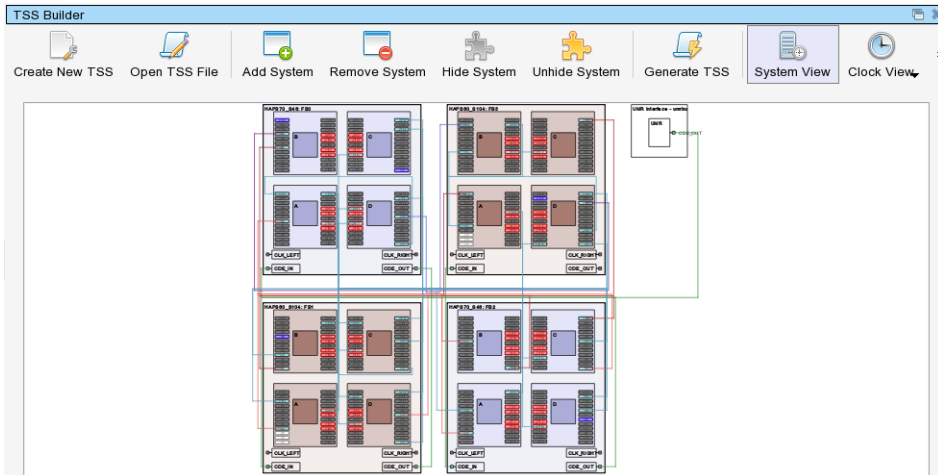
## System View



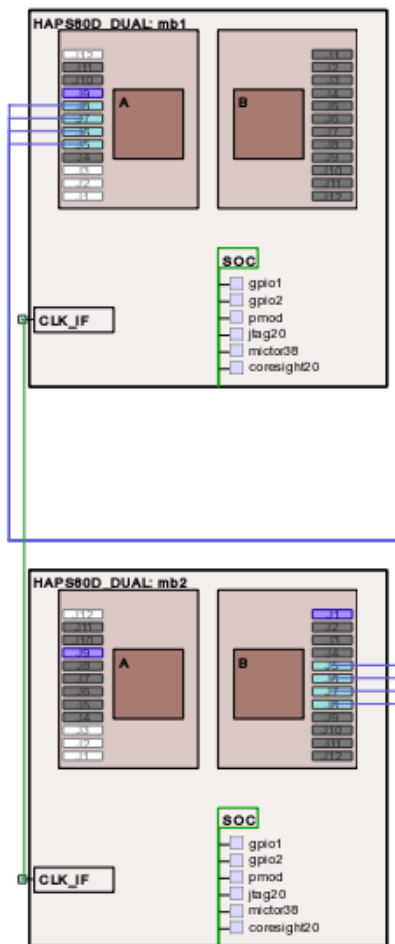
Provides a graphical representation of all systems in the TSS file. The TSS Builder window displays the HAPS systems, CDE connections, and clock connections. You can modify connections, rename systems, and add or remove systems from this view.

The following is an example of system view with HAPS-70 and HAPS-80 systems, and a UMRBUS\_IF:

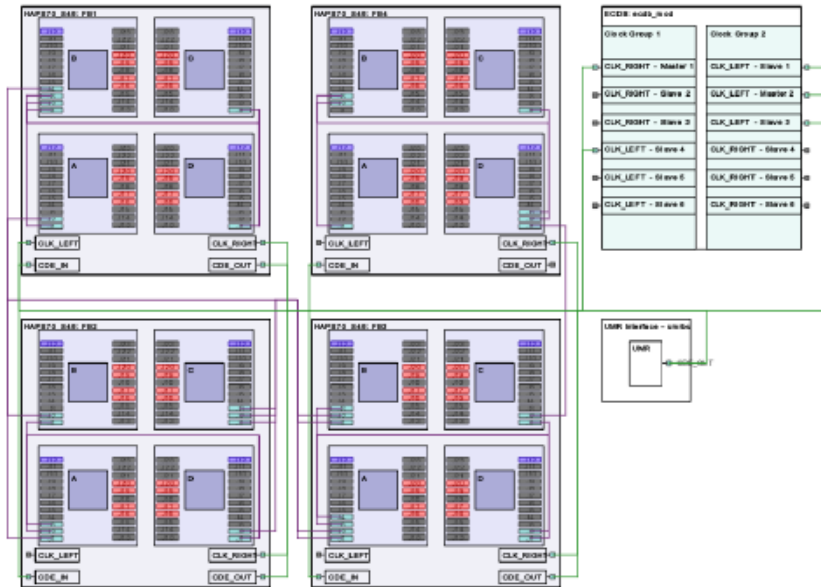




The following is an example of system view with two HAPS-80D systems and a UMRBUS\_IF.



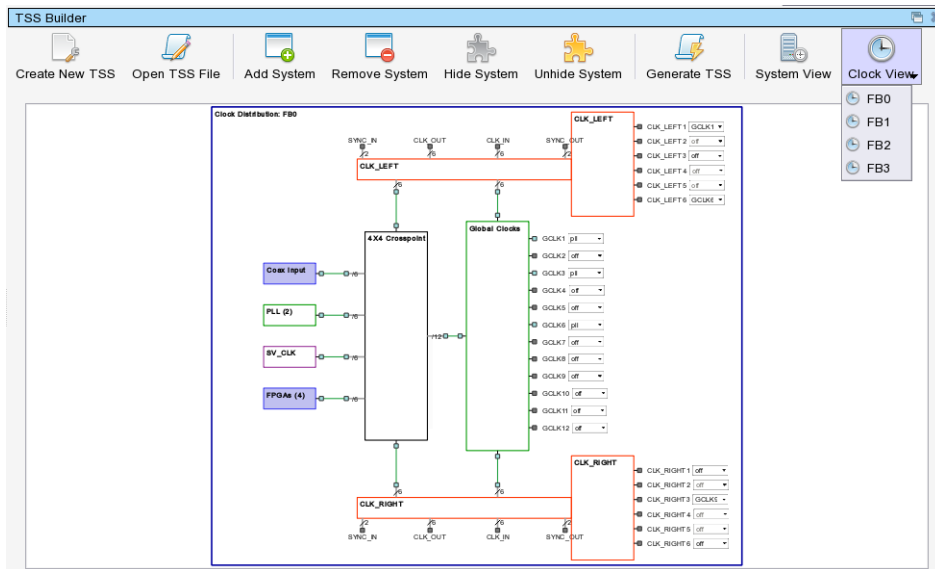
The following is an example of system view with HAPS systems connected to an ECDB:



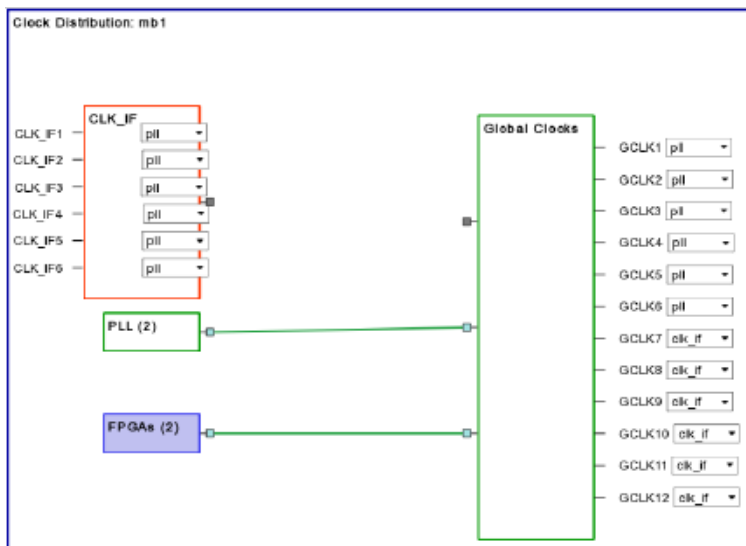
## Clock View



Provides a graphical view of the clock details. You can modify the clock details from this view. Use the drop-down menu to select the HAPS system to modify the clock details.



The following is an example of the clock view for a HAPS-80D system:



## Connection Table



Displays connections between the systems in a table format.

TSS Builder				
<div>  Create New TSS            Open TSS File            Add System            Remove System            Hide System            Unhide System            Generate TSS            System View            Clock View            Connection Table            Daughter Board Table         </div>				
	Name	Port 1	Port 2	Cable Used
1	B6_C12	FB0.B6	FB2.C12	CON_CABLE_200_HT3
2	B9_D10	FB0.B9	FB3.D10	CON_CABLE_50_HT3
3	D7_B11	FB0.D7	FB3.B11	CON_CABLE_100_HT3
4	A6_B7	FB0.A6	FB2.B7	CON_CABLE_150_HT3
5	C10_A10	FB0.C10	FB1.A10	CON_CABLE_25_HT3
6	C12_D11	FB0.C12	FB2.D11	CON_CABLE_25_HT3
7	D20_A20	FB0.D20	FB3.A20	CON_CABLE_25_HT3
8	A17_D17	FB0.A17	FB1.D17	CON_CABLE_25_HT3
9	C16_D16	FB0.C16	FB2.D16	CON_CABLE_25_HT3
10	C15_C12	FB2.C15	FB1.C12	CON_CABLE_25_HT3
11	D8_C17	FB2.D8	FB1.C17	CON_CABLE_25_HT3
12	B1_C12	FB1.B1	FB3.C12	CON_CABLE_200_HT3
13	D12_D13	FB1.D12	FB3.D13	CON_CABLE_25_HT3
14	A23_D1	FB0.A23	FB1.D1	CON_CABLE_25_HT3
15	B13_A12	FB2.B13	FB3.A12	CON_CABLE_200_HT3
16	A8_C7	FB2.A8	FB1.C7	CON_CABLE_25_HT3
17	B4_D1	FB1.B4	FB3.D1	CON_CABLE_25_HT3
18	A4_C1	FB1.A4	FB2.C1	CON_CABLE_150_HT3

Each row in the table represents a connection in the system configuration. Name of the connection, ports connected, and the cable used for the connection is displayed in this table.

You may create new connections by entering connection details directly to the table. Connections entered in the table gets reflected in the graphical view.

## Daughter Board Table



Displays the daughter boards details in a table format.

TSS Builder				
<div>  Create New TSS            Open TSS File            Add System            Remove System            Hide System            Unhide System            Generate TSS            System View            Clock View            Connection Table            Daughter Board Table         </div>				
	Daughter Board	Given Name	Connected to System	Ports
1	CON_HT3	db1	FB0 FB2	JX1 : FB0.A1 JX2 : FB2.B2 All Ports are connected.
2	DTD_MEM	db2	FB0 FB1 FB2	JX1 : FB0.A3 JX2 : FB1.A3 JX3 : FB2.C2 All Ports are connected.
3	BREAKOUT_HT3	db0	FB2 FB0 FB3	J1 : FB2.B3 J2 : FB0.C4 JX1 : FB3.C5 All Ports are connected.
4				

Each row in the table represents a daughter board. Daughter board name, given name, HAPS system to which the board is connected, ports used by the daughter board, and notes are displayed for each board.

### Zoom In



Magnifies the display within the TSS Builder window.

### Zoom Out



Reduces the size of the display within the TSS Builder window.

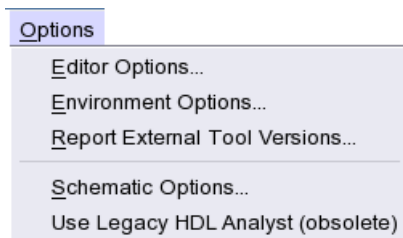
### Fit Screen



Fits the system configuration display to the exact window size in the TSS Builder Window.

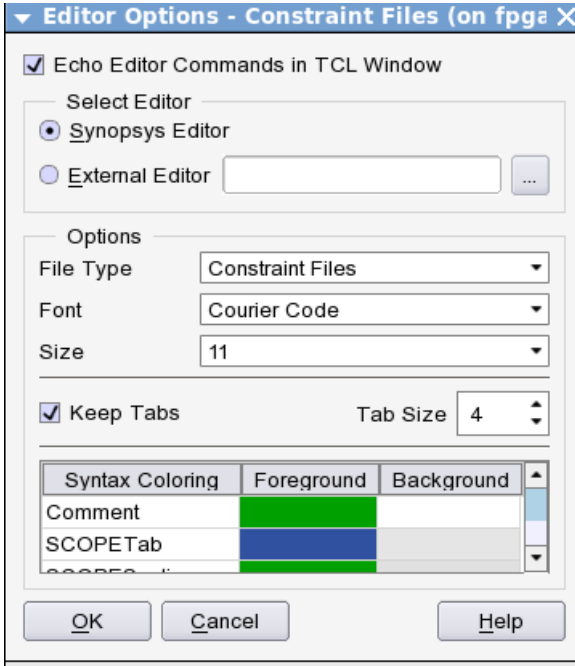
## Options Menu

The Options menu sets editor, environment, and analyst options for the tool.



Editor Options	Opens the Editor Options - Constraints Files dialog to allow the text editor option settings to be modified for the session. For more information on the Synopsys editor, see <a href="#">Editor Options - Constraints Files Dialog Box</a> , on page 71.
Environment Options	Opens the Environment Variables Options dialog to allow system environment variables to be set for the ProtoCompiler. See <a href="#">Environment Variables Options Dialog Box</a> , on page 72.
Report External Tool Versions	Opens the External Tool Versions window showing the supported versions and install environment of external tools.
Schematic Options	Opens dialog boxes for setting options of the HDL Analyst schematics. For more information, see either: <ul style="list-style-type: none"> <li>• <a href="#">HDL Analyst Options Dialog Box</a> , on page 72</li> <li>• <a href="#">Standard HDL Analyst Options Dialog Box</a> , on page 74.</li> </ul>

## Editor Options - Constraints Files Dialog Box

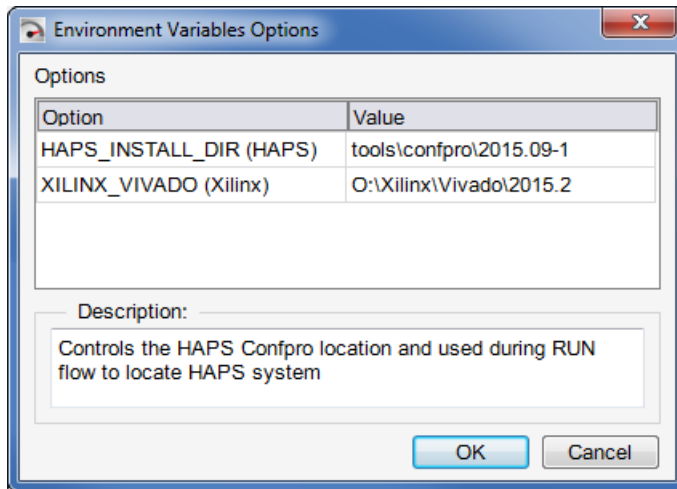


Command	Description
Synopsys Editor	Sets the Synopsys text editor as the default text editor.
External Editor	Uses the specified external text editor program to view text files from within ProtoCompiler. The executable specified must open its own window for text editing; files opened with an external editor cannot be crossprobed. See Using an External Text Editor for a procedure.
File Type	Defines the text editor preferences for the supported file types from the drop-down list.
Font	Selects the font to be used with the text editor from the drop-down list.

Command	Description
Font Size	Defines the font size to use with the text editor from the drop-down list.
Keep Tabs/Tab Size	Defines the tab setting for the text editor.
Syntax Coloring	Defines foreground or background syntax coloring to use with the text editor using the standard color pallet.

## Environment Variables Options Dialog Box

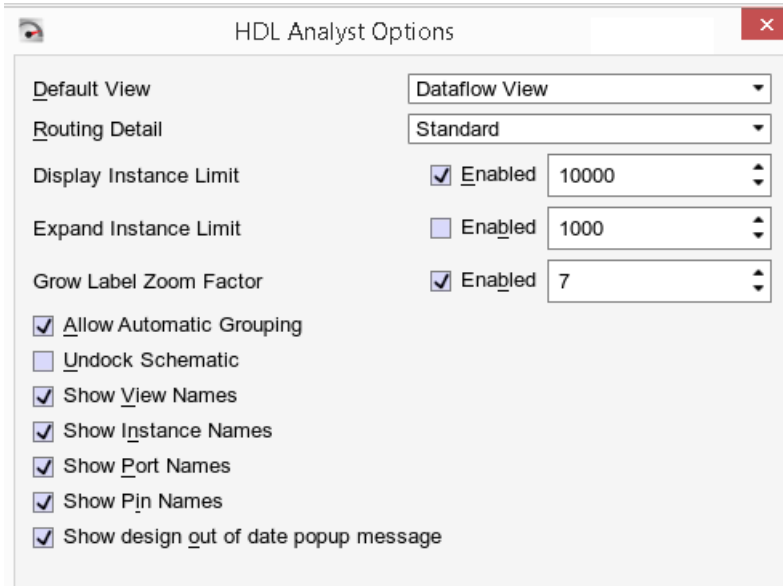
The Environment Variables Options dialog box lists the available environment variables and their current settings. Click the variable name in the Option column to display the description.



## HDL Analyst Options Dialog Box

Select Options->Schematic Options to display a dialog box where you define preferences for the HDL Analyst schematic. For details see [Setting the Schematic Preferences, on page 103](#) in the *User Guide*.





The following options are on the HDL Analyst Options panel.

Field/Option	Description
Default View	Specify how you want the schematic views to display: <ul style="list-style-type: none"> <li>• Compact View - Displays cluster modules based on interconnectivity, for example, highly interconnected modules are grouped closer together. The compact view is expected to take longer to load, however, the view will be useful when, for example, manually partitioning a design.</li> <li>• Clocks View - Displays all sequential elements connected to clock nets so that clocks in the design can be debugged.</li> <li>• Dataflow View - Displays objects from a left to right datapath flow. This is the default.</li> </ul>
Routing Detail	<ul style="list-style-type: none"> <li>• Specify how the tool determines the detailed routing for the design: Standard - This is the default.</li> <li>• Quick - Direct net connections</li> </ul>
Display Instance Limit	When enabled, uses the specified limit to display instances. The default is 10000.
Expand Instance Limit	When enabled, uses the specified limit to expand instances. The default is 1000.

Field/Option	Description
Grow Label Zoom Factor	Specify a zoom factor for labels displayed in the schematic view. Select a value between 1 and 10, where labels are shown increasing in size respectively. Changes will appear in the next opened schematic view. The default is 2.
Allow Automatic Grouping	When enabled, automatic grouping is performed.
Show View Names	When enabled, module names are displayed.
Undock Schematic	When enabled, schematic is not docked.
Show Instance Names	When enabled, instance names are displayed.
Show Port Names	When enabled, port names are displayed.
Show Pin Names	When enabled, pin names are displayed.
Show design out of date popup message	When enabled, shows the design out of date popup message.
Restore Defaults	Click this button to reset all options to their defaults.

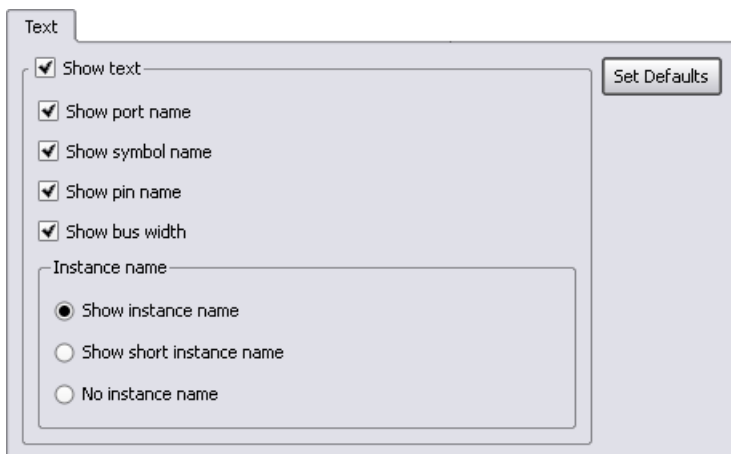
## Standard HDL Analyst Options Dialog Box

Select Options->HDL Analyst Options to display the HDL Analyst Options dialog box, where you define preferences for the HDL Analyst schematic views (compile and map views). For details, see [Setting Schematic Preferences, on page 164](#) in the *User Guide*.

For information about the options, see the following, which correspond to the tabs on the dialog box:

- [Text Panel](#), on page 75
- [General Panel](#), on page 76
- [Sheet Size Panel](#), on page 79
- [Visual Properties Panel](#), on page 81

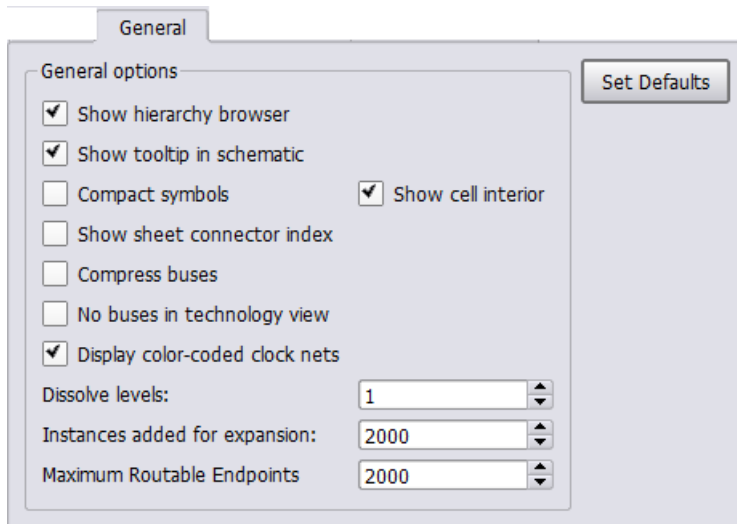
## Text Panel



The following options are in the Text panel.

Field/Option	Description
Show text	Enables the selective display of schematic labels. Which labels are displayed is governed by the other Show * features and Instance name, described below.
Show port name	When enabled, port names are displayed.
Show symbol name	When enabled, symbol names are displayed.
Show pin name	When enabled, pin names are displayed.
Show bus width	When enabled, connectivity bit ranges are displayed near pins (in square brackets: [ ]), indicating the bits used for each bus connection.
Instance name	Determines how to display instance names: <ul style="list-style-type: none"><li>• Show instance name</li><li>• Show short instance name</li><li>• No instance name</li></ul>
Set Defaults	Set the dialog box to display the default values.

## General Panel



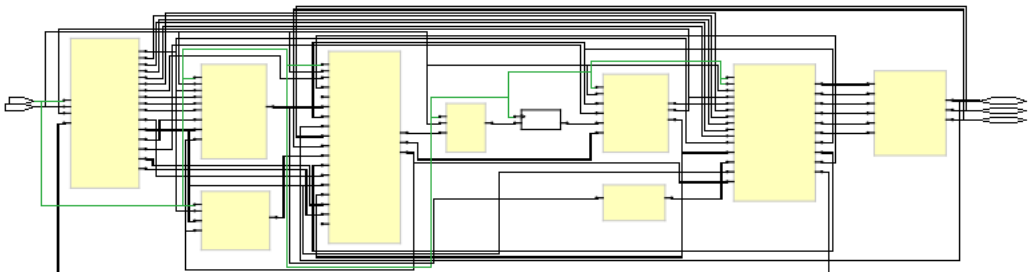
The following options are in the General panel.

Field/Option	Description
Show hierarchy browser	When enabled, a hierarchy browser is present at the left pane of schematic view.
Show tooltip in schematic	When enabled, displays tooltips that when hovering over objects in the schematic view.
Compact symbols	When enabled, symbols are displayed in a slightly more compact manner, to save space in schematics. When this is enabled, Show cell interior is disabled.
Show cell interior	When enabled, the internal logic of cells that are technology-specific primitives (such as LUTs) is shown in mapped views. This is not available if Compact symbols is enabled.
Show sheet connector index	When enabled, sheet connectors show connecting sheet numbers.
Compress buses	When enabled, buses having the same source and destination instances are displayed as bundles, to reduce clutter. A single bundle can connect to more than one pin on a given instance. The display of a bundle of buses is similar to that of a single bus.

Field/Option	Description
No buses in technology view	When enabled, buses are not displayed, they are only indicated as bits in a mapped view. This applies only to flattened views, not to hierarchical views that have been flattened.
Display color-coded clock nets	Displays clock nets in the HDL Analyst View with the color green. See <a href="#">Color-coded Clock Nets</a> , on page 77.
Dissolve levels	The number of levels to dissolve, during HDL Analyst->Dissolve Instances.
Instances added for expansion	The maximum number of instances to add during any operation (such as HDL Analyst->Hierarchical->Expand) that results in a <i>filtered</i> schematic. When this limit is reached, you are prompted to continue adding more instances.
Maximum Routable Endpoints	Specifies the maximum number of endpoints for nets, which the synthesis tool routes to their explicit connection endpoints in the design to improve HDL Analyst performance.  The default value is set to 2000. You can use this option to change this value. For more information, see <a href="#">Results of Maximum Routable Endpoints in the HDL Analyst View</a> , on page 78.

## Color-coded Clock Nets

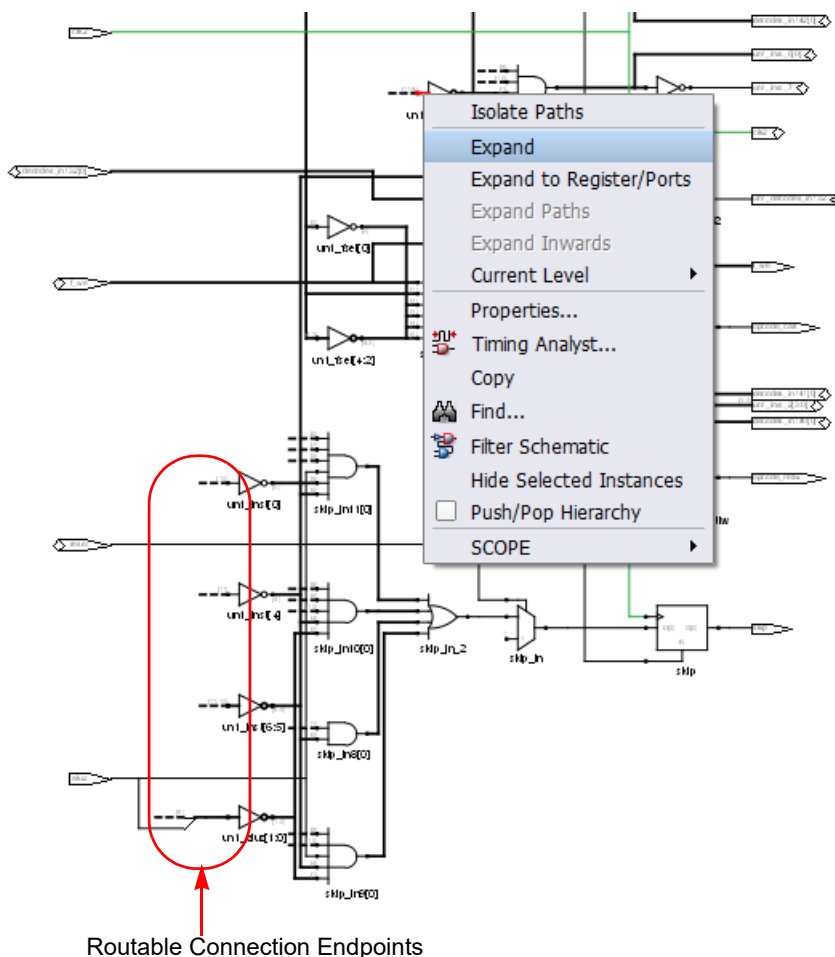
Clock nets are displayed with the color green in the schematic views.



## Results of Maximum Routable Endpoints in the HDL Analyst View

Use the Maximum Routable Endpoints option to specify the maximum number of endpoints for nets in the design to be explicitly routed to their connection endpoints. When you adjust the default value of 2000 sufficiently, improvements in performance can be seen in the HDL Analyst tool.

When the number of connection endpoints routed for the design has been reduced, you will see dashes (---) for these endpoints in the schematic view. Note that you can still select these endpoints and perform any viable operation for these nets as shown in the compile view below.

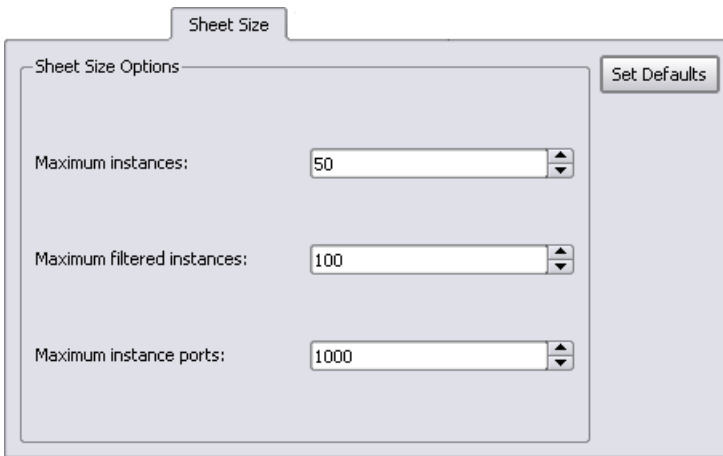


---

**Note:** Occasionally, the software does not route nets. You will see dashes (---) for these endpoints in the schematic view. Note that you can still select these nets and perform any viable operation for them.

---

## Sheet Size Panel



The Sheet Size Panel is a dialog box with a tab labeled "Sheet Size". It contains a "Sheet Size Options" section with three spinners: "Maximum instances:" set to 50, "Maximum filtered instances:" set to 100, and "Maximum instance ports:" set to 1000. A "Set Defaults" button is located in the top right corner.

Option	Value
Maximum instances:	50
Maximum filtered instances:	100
Maximum instance ports:	1000

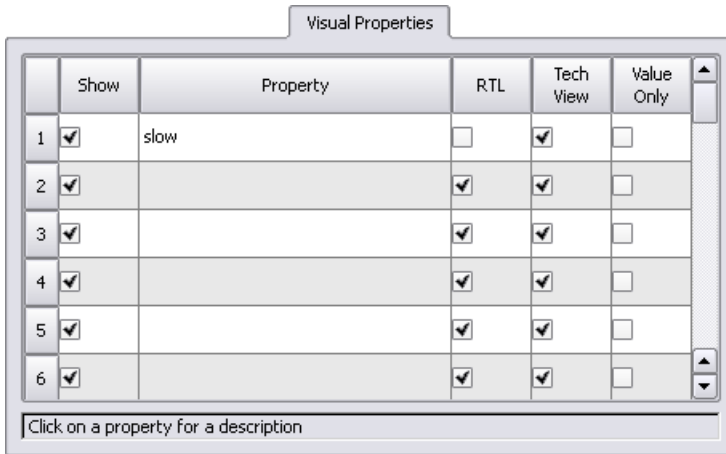
The following options are in the Sheet Size panel.

Maximum instances	Defines the maximum number of instances to display on a single sheet of an unfiltered schematic. If a given hierarchical level has more than this number of instances, then it will be partitioned into multiple sheets.
Maximum filtered instances	<p>Defines the maximum number of instances to display on a filtered schematic sheet, at any visible hierarchical level. This limit is applied recursively, at each visible <i>level</i>, when the sheet itself is a level or when each transparent instance is a level (even when within another transparent instance).</p> <p>Whenever a given level has more child instances inside it than the value of Filtered Instances, it is divided into multiple sheets.</p> <p>At each level except the sheet itself, an additional margin of allowable child instances is added to the Maximum filtered instances value, increasing its effective value. This means that you can see more child instances than Maximum filtered instances itself implies.</p> <p>The Maximum filtered instances value must be at least the Maximum instances value.</p>
Maximum Instance Ports	Defines the maximum number of instance pins to display on a schematic sheet.



## Visual Properties Panel

Controls the display of the selected property in open HDL Analyst views. The properties are displayed as colored boxes on the relevant objects. To display these properties, the View->Visual Properties command must also be enabled. For more information about properties, see [Viewing Object Properties, on page 158](#) in the *User Guide*.



The following options are in the Visual Properties panel.

Show	Toggles the property name and value is displayed in a color-coded box on the object.
Property	Sets the properties to display.
RTL	Enables or disables the display of visual properties in the compile view.
Tech View	Enables or disables the display of visual properties of in the mapped view.
Value Only	Displays only the value of an item and not its property name.

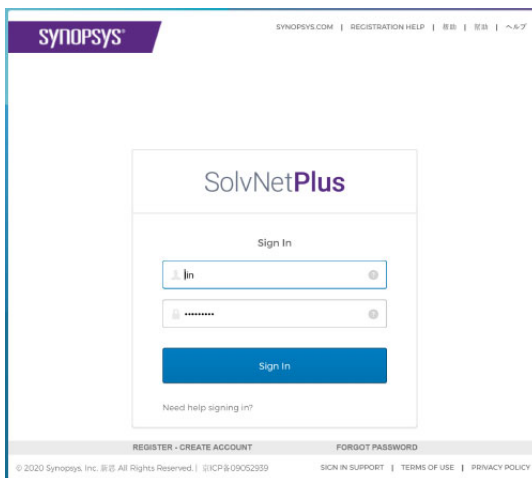
## Tech-Support Menu

The Tech-Support menu contains information and the actions you can take when you encounter problems running your designs or working with the Synopsys FPGA Implementation products.

Command	Description
Web Support	<p>Opens the Synopsys SolvNetPlus Support page from where you can:</p> <ul style="list-style-type: none"><li>• Log on to SolvNetPlus to request Synopsys technical support.</li><li>• Access the Synopsys Products, Downloads, Training, and Documentation pages that have links to product information.</li></ul> <p>See <a href="#">Web Support Command</a> , on page 82 for more information.</p>

## Web Support Command

Through the Synopsys SolvNetPlus Support page, you can access Products, Downloads, Training, and Documentation pages on the Synopsys website as well as submit requests for technical support through SolvNetPlus. To use SolvNetPlus, you must have a user account. You are prompted for your SolvNetPlus account when you select Tech-Support->Web Support from the main menu.



## Web Menu

This menu contains commands that access up-to-date information from Synopsys Support.

Command	Description
Go to SolvNet	Opens the home page for the Synopsys SolvNetPlus Search support. This website contains links to useful technical information. You can search for new or updated articles or documentation, such as application notes, white papers, release notes, and other user-oriented documentation.
Go to Training Center	Opens the Synopsys training web page for Synopsys products. Synopsys offers both online web-based training courses, as well as classroom training courses taught by Synopsys personnel. Select the FPGA Implementation courses from the drop-down menu.
Synopsys Home	Opens the Synopsys home web page for Synopsys products.
FPGA Implementation Tools	Opens the Synopsys FPGA design solution web page for Synopsys FPGA products. You can find information about the full line of Synopsys FPGA Implementation products here.

## Help Menu

The Help menu provides a level of user help from within the graphical window. The following table describes the individual Help menu selections and their functions.

Command	Description
Help Topics	Displays hyperlinked online help for the product.
HAPS Design Advisor	Displays the HAPS Design Advisor which provides guidance, advice, and best practices for prototyping using ProtoCompiler.
How to Use Help	Displays help on how to use Synopsys online help.
PDF Documents	Displays an Open dialog box with hyperlinked PDF documentation on the product including user guides and reference manuals. An Adobe Acrobat Reader is required to view the PDF files.

Command	Description
Error Messages	Displays help on the message viewer.
TCL	Displays help for Tcl commands.
License Agreement	Displays the Synopsys FPGA software license agreement.
Floating License Usage	Specifies the number of floating licenses available and the number of licenses currently in use.
About this program	<p>Displays the About dialog box, showing the synthesis tool product name, license expiration date, customer identification number, version number, and copyright.</p> <p>Clicking the Versions button in the About dialog box displays the Version Information dialog box, listing the installation directory and the versions of all the synthesis tool compiler and mapper programs.</p>

# Icons

The menu bar icons are described in the following subsections. The individual icons are enabled for specific database conditions; a number of the icons duplicates functions available both from the drop-down menus and the Database View window.

## Create Database

Opens the Create Database dialog where you enter a database directory location and a name for the database. Clicking OK opens the new database in the Database View window. The icon performs the same function as the New Database Setup button in the initial Database Design Flow window.

## Open Database

Opens a browser where you can select an existing database directory. Clicking Choose opens the selected database in the Database Design Flow window. The icon performs the same function as the Open Database button in the initial Database Design Flow window.

## Close Database

Closes the current database and returns to the initial Database Design Flow window.

## Edit Options

Opens the Options Editor dialog to allow you to change the default option settings. The options are listed with their current value settings. Options with enable/disable (Boolean) values use a check box, and options with string values use a drop-down menu. Changing an option value in the dialog does not change the value of a corresponding option defined in a constraint file. For a description of the editor, see [Options Editor, on page 134](#).

## Edit File List

Opens the initial File List Editor dialog box to create or edit a file list. For a description of the editor, see [File List Editor, on page 128](#).

## Edit Identify Design Constraint

Prompts for an IDC file by opening the Edit Instrumentation dialog box. Selecting a previous instrumentation file and clicking Open opens the file in the instrumentor GUI or a text editor. For information on editing IDC files, see [Instrumenting the Design for Debug, on page 641](#) in the User Guide.

## Edit FPGA Constraint

Opens the FPGA constraint editor, a spreadsheet-like editor with a number of panels for entering and managing timing constraints and synthesis attributes. For more information, see [Specifying FDC Constraints, on page 20](#) in the *Compiler and Mapper Guide*.

## Edit Partition Constraints

Opens the PCF (Partition Constraint File) interactive editor for PCF files. The editor is available from an active pre-partition, partition, or system route database state. For additional information, see [PCF Graphical Editor, on page 116](#) (see [Chapter 3, Partition Constraint File Tcl Commands](#) in the Command Reference and [Defining Constraints in PCF Files, on page 241](#) in the *ProtoCompiler User Guide*). To open the PCF (Partition Constraint File) editor, click the Edit Partition Constraints icon.

## Timing Report View

Opens the Timing Report View, which displays timing reports, lets you view and query critical timing paths, and correlate your timing results, see [Timing Report View, on page 40](#) and [Viewing and Correlating Results with the Timing Report View, on page 535](#) in the *ProtoCompiler User Guide*.

## Timing Analyst

The Timing Analyst or timing analyzer lets you generate custom reports for paths, see [Timing Analyst, on page 48](#).

## View Schematic

Opens a schematic, if available, for the current database state in the adjacent window (*designName* tab). The schematic view displayed depends on the current database state; when multiple schematics are available, the adjacent scroll bar selects the schematic view from the corresponding database state.

## View Reports



Opens the log file for the current database state in the adjacent window as a Report View tab. This window displays a hierarchy of log files in chronological order on the left and the log file content on the right. For additional information, see [Checking Reports and Log Files, on page 512](#) in the User Guide and [Chapter 5, Inputs, Reports and Log Files](#) in the Reference.

## View Database Summary



Opens a summary page detailing the results from each completed database state in a Summary window. For additional information, see [Summary View, on page 97](#).

## View Messages



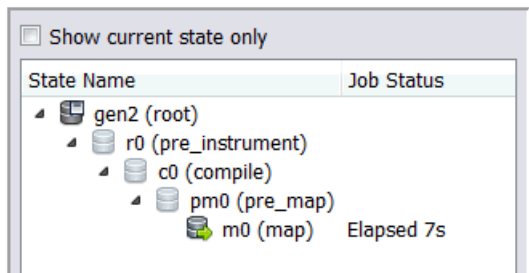
Opens the Report Messages View which lists the individual messages concerning a database state, see [Manipulating Message Display and Reporting, on page 551](#) in the *User Guide*.

## HAPS Design Advisor

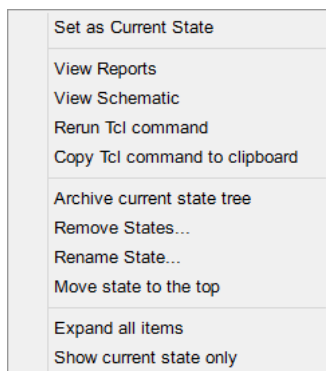
The HAPS Design Advisor provides guidance, advice and best practices for prototyping using ProtoCompiler.

## State Tree View Operations

The individual databases appear in the State Name area of the Database View. The databases appear in flow sequence and are indented as shown in the following figure with the active database highlighted (m0 in the figure).



Positioning the cursor over a database state and right clicking displays a popup menu with the following controls.



### Set as Current State

Sets the selected database state to the current database state. All subsequent database states are removed from the display. This selection is equivalent to double-clicking on a database state.

### View Reports

Opens a Report View that includes all of the reports or logs appropriate for the current database state beginning with the first report. The report hierarchy on the left lists all of the available reports.

### View Schematic

Opens a schematic for the current database state in a separate window. The schematic view displayed depends on the current database state.



**Rerun Tcl command**

Repeats execution of the corresponding Tcl command. Applying this command to an earlier database state repeats the command, but removes all subsequent database states.

**Copy Tcl command to clipboard**

Copies the corresponding run command responsible for generating the database state to the clipboard. The complete command syntax can then be added to a Tcl file for scripting.

**Archive current state tree**

Includes all files related to the current database state in the archive. For additional information on archiving, see [Archiving and Unarchiving Databases](#), on page 66 in the *User Guide*.

**Remove States**

Removes the selected database state and all subsequent database states from the display. You are prompted to confirm the removal.

**Rename State**

If you have many states of the same type, you can rename a state to keep track of the best one.

**Move state to the top**

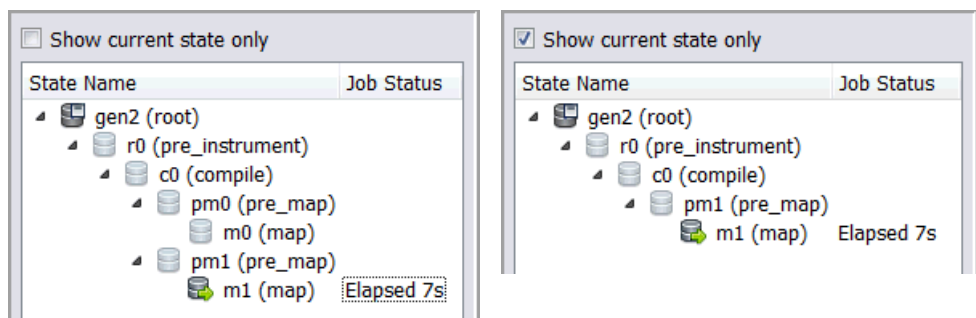
Use this option to reorganize the database tree by moving the most useful runs to the top.

**Expand all items**

Displays any database states that exist below the current database state.

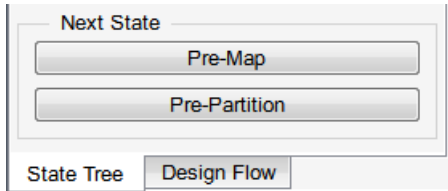
**Show current state only**

Displays only the active database states by removing any inactive database states from the display.



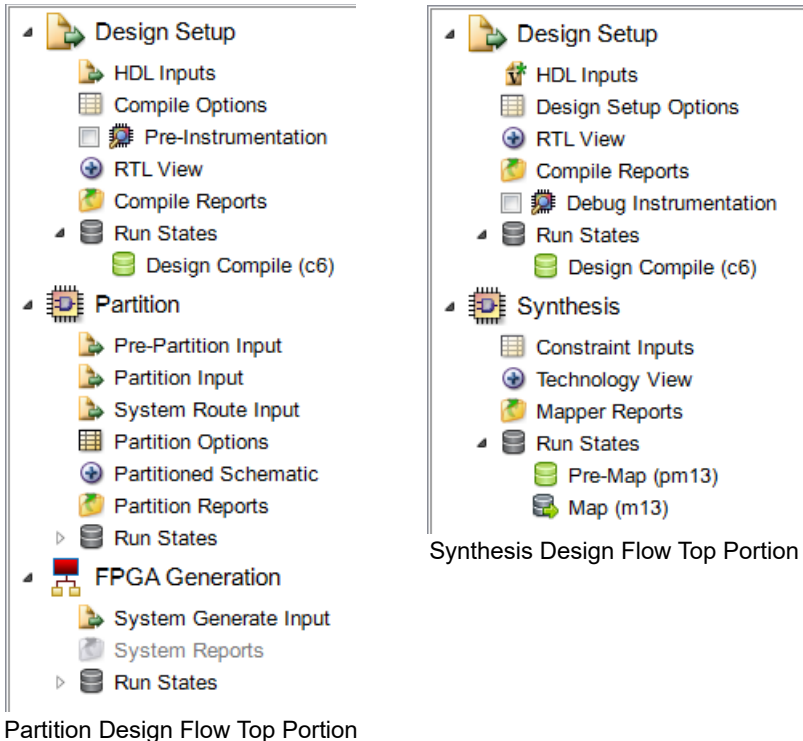
## State Tree Bottom Menu

The portion of the menu at the bottom of the State Tree view displays the next available database state or states available for selection (the button labels change according to the current state). Clicking on a button opens the corresponding task dialog box (see [Task Dialog Boxes, on page 100](#)).



# Design Flow View Operations

The Design Flow view is a template for design definition and configuration. Two unique views are available as determined by the Flow Type drop-down menu selection at the bottom of the view.



## Partition Design Flow

The top portion of the partition design flow view is divided into three major areas: Design Setup, Partition, and FPGA Generation.

## Design Setup

HDL Inputs	Opens a file list editor window to define the HDL design files for the design (see <a href="#">File List Editor</a> , on page 128).
Compile Options	Opens the options editor window to define the options settings for the design (see <a href="#">Options Editor</a> , on page 134).
Pre-Instrumentation	Enables design debug functions when checked (adds a debug run state to the flow).
RTL View	Opens an available RTL view.
Compile Reports	Displays report window of log files in chronological order on the left and the log file content on the right.
Run States	Lists the available run states associated with design setup.

## Partition

Pre-Partition Input	Opens the Database Files window to allow constraint file selection for the pre-partition database state.
Partition Input	Opens the Database Files window to allow constraint file selection for the partition database state.
System Route Input	Opens the Database Files window to allow constraint file selection for the system-route database state
Partition Options	Opens the options editor window to define the options settings associated with the partitioning flow.
Pre-partitioned Schematic	Opens a schematic view of the pre-partitioned schematic.
Partitioned Schematic	Opens a schematic view of the partitioned schematic.
Partition Reports	Displays report window of log files up to the partitioned database on the left and the log file content on the right.
Run States	Lists the available run states associated with partitioning.

## FPGA Generation

System Generate Input	Opens the Database Files window to allow constraint file selection for the system-generate database state.
System Reports	Displays report window of log files up to the system generation database on the left and the log file content on the right.
Run States	Lists the available run states associated with FPGA generation.

## Synthesis Design Flow

The top portion of the synthesis design flow view is divided into two areas: Design Setup and Synthesis.

### Design Setup

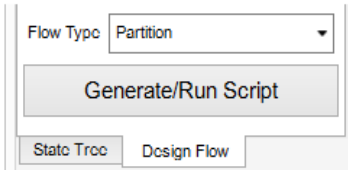
HDL Inputs	Opens a file list editor window to define the HDL design files for the design (see <a href="#">File List Editor</a> , on page 128).
Design Setup Options	Opens the options editor window to define the options settings for the design (see <a href="#">Options Editor</a> , on page 134).
RTL View	Opens an available RTL view.
Compile Reports	Displays report window of log files in chronological order on the left and the log file content on the right.
Debug Instrumentation	Enables design debug functions when checked (adds a debug run state to the flow).
Run States	Lists the available run states associated with design setup.

### Synthesis

Constraint Inputs	Opens the Database Files view.
Technology View	Opens an available post-map view.
Mapper Reports	Displays report window of log files in chronological order on the left and the log file content on the right.
Run States	Lists the available run states associated with design synthesis.

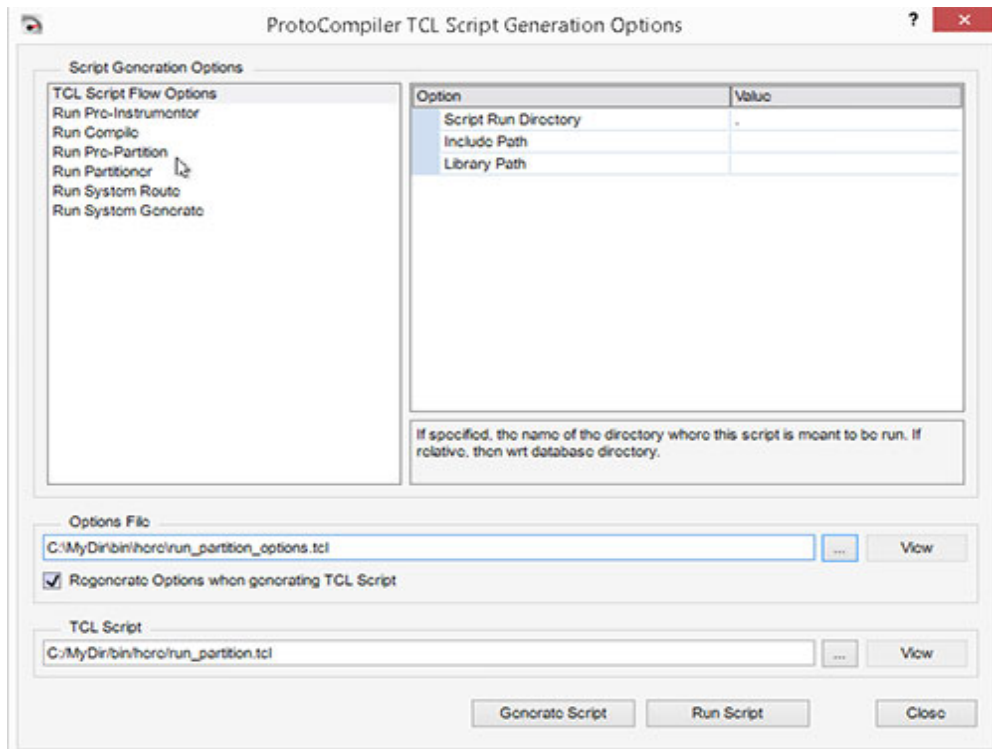
## Design Flow Bottom Menu

The portion of the menu at the bottom of the Design Flow view selects the flow type (Synthesis or Partition). After selecting the flow and completing the corresponding parameter entries in the upper portion of the Design Flow view, clicking the Generate/Run Script button opens the Design Flow Popup window.



## Design Flow Popup Menu

Right clicking in the Design Flow view brings up a Generate Script Options selection which, when selected, opens the ProtoCompiler TCL Script Generation Options dialog box.



Script Generation Options	Lists the available options to be scripted. Options are displayed in flow order and include both state names and check boxes to include the selected option.
Option Description	Describes the option function.
Options File	Lists the path to the location of the generated script.
TCL Script	Specified name for the generated script.
View	Opens generated script in text editor.
Generate Script	Generates the script according to the script generation options.
Run Script	Runs generated script in TCL window.

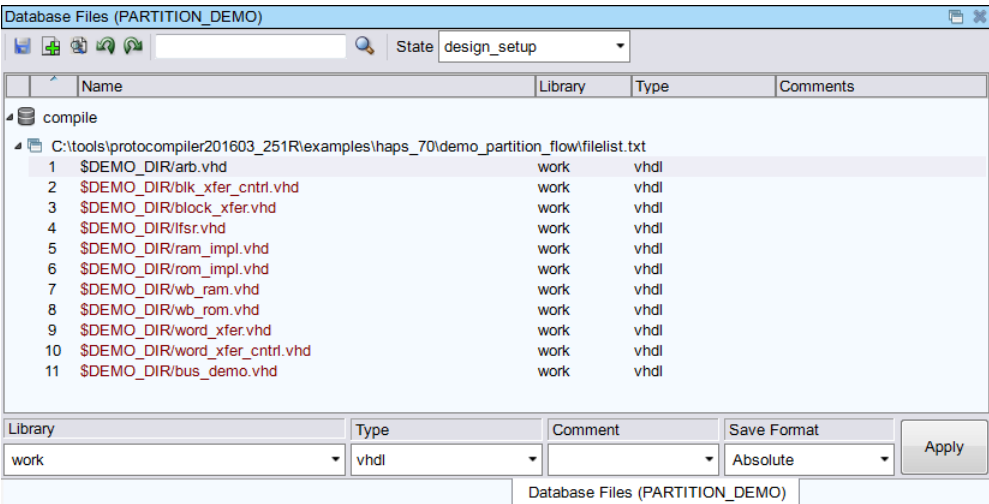
# Secondary Views

Individual secondary views are associated with various operations which, when enabled, open over the TCL Script window. These views can overlap one another and are redisplayed by selecting their corresponding tabs across the bottom on the window. The following secondary views are described in this section:

- [Database Files View](#), on page 96
- [Summary View](#), on page 97
- [Report Messages View](#), on page 97

## Database Files View

The Database Files view opens directly from specific selections in the Design Flow view. This view, with the exception of the State drop-down menu, is essentially the same as the file editor described under [File List Editor, on page 128](#). To use the view, click the Add Files icon and select the files from the Select Files for Design Setup dialog box.



The State drop-down menu lists the acceptable file types for the selected state.



## Summary View

The summary view provides an overview of the status in an at-a-glance format of the reports generated and associated messages.

Database Details				
Database Name	SYSTEM_IP_BUTTON_LED_DEMO			
Current State Tree	c0pp0:pal0:sr0:sg0			
Path	C:\Users\lindan\Desktop\protocompiler201609sep1_06SR\bin\SYSTEM_IP_BUTTON_LED_DEMO			
Target Device	protol: Synopsys HAPS-80 : XC7VU440			
cd (compile)				
Command	run compile -sclist C:\Users\lindan\Desktop\protocompiler201609sep1_06SR\examples\haps_80\demo_system_ip_button_led_flow\filelist.txt -top_module up_counter_load -vlog_std sysv			
Run Time	2 seconds			
Run Status	Complete			
📄 Reports				
Export Reports				
Compile Log: <a href="#">up_counter_load_compiler.srl</a>	5	0	0	02:00:54 Oct 17 2016
Multi-srs Generator Log: <a href="#">up_counter_load_multi_srs_gen.srl</a>	1	0	0	02:00:54 Oct 17 2016
Total	6	0	0	


Clicking on a Reports category in one of the database summaries lists the available reports in the Report Messages view (see [Report View, on page 157](#)) and clicking on a populated message type (notes, warnings, or errors) opens the associated Report Messages view (see [Report Messages View, on page 97](#)) and [Handling Errors and Warnings, on page 550](#) in the *User Guide*.

To export reports click on the link Export Reports and select the desired directory in the dialog. Select a directory to Export Reports, then click Choose.

Exports
<a href="#">export runtime -path HSTDM_FLOW_DEMO_RUNTIME</a>

All data that has been exported from a state using the **export** command (see [export, on page 54](#) in the Command Reference) is listed under the Exports tab. Clicking on a link opens the Export Information view, where all associated files are listed.

## Report Messages View

To open the Report Messages view, click on the View Messages icon (  ) or select View->Message View from the menu to list the individual messages that apply to the corresponding database state. Clicking on a blue highlighted message ID displays the message description from the *Messages Reference*.

Report Messages: c0

6 warnings, 28 notes

Find:

Set Filter...

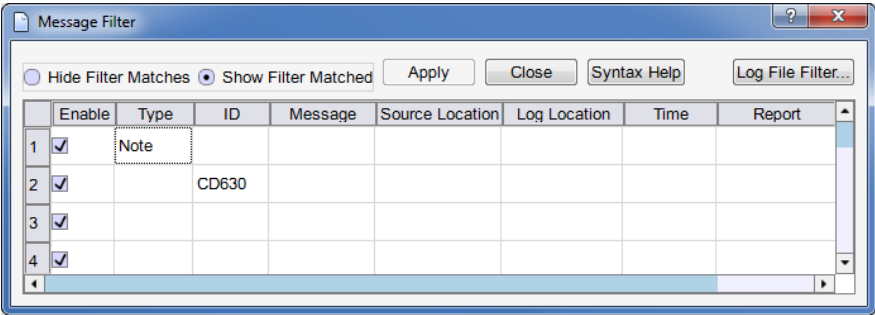
Apply Filter

Group Common IDs

Type	ID	Message	Source Location	Log Location	Time	Report
Warning	5	CD638 Signal clkfx is undriven. Either assign th...	bus_demo.vhd	bus_demo_c...	12:31:14...	Compile Log
Warning		CL246 Input port bits 9 to 5 of addr(9 downto 0) ...	rom_impl.vhd (31)	bus_demo_c...	12:31:14...	Compile Log
Note		Running in 64-bit mode	-	bus_demo_m...	12:31:16...	Multi-srs Generator Log
Note		Running in 64-bit mode	-	bus_demo_c...	12:31:14...	Compile Log
Note		Running in 64-bit mode	-	bus_demo_c...	12:31:14...	Compile Log
Note		Running compiler in Fast Synthesis mode	-	bus_demo_c...	12:31:14...	Compile Log
Note		Running in 64-bit mode	-	bus_demo_c...	12:31:14...	Compile Log
Note	14	CD630 Synthesizing work.word_xfer.beh.	-	bus_demo_c...	12:31:14...	Compile Log
Note	5	CD233 Using sequential encoding for type states	-	bus_demo_c...	12:31:14...	Compile Log
Note		Top entity is set to bus_demo.	bus_demo.vhd (32)	bus_demo_c...	12:31:14...	Compile Log
Note	2	CL134 Found RAM mem, depth=256, width=8	ram_impl.vhd (40)	bus_demo_c...	12:31:14...	Compile Log
Note		CD720 Setting time resolution to ns	std.vhd (123)	bus_demo_c...	12:31:14...	Compile Log

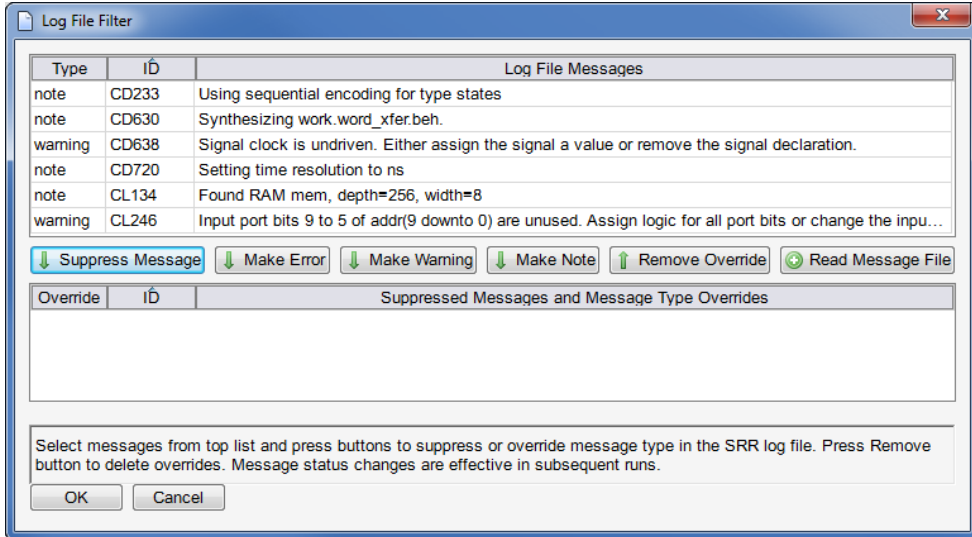
Message Filter Dialog Box

The Message Filter dialog box is used to filter out unwanted messages. Click Set Filter in the Report Messages view to display the dialog box. Set the columns to reflect the filter criteria. Clicking on the Log File Filter button brings up the Log File Filter dialog box to allow individual messages to be suppressed or promoted/demoted in importance for the next run.



Log File Filter Dialog Box

The Log File Filer dialog box is the primary control for changing a message priority or suppressing a message for the next run. For additional information, see *Manipulating Message Display and Reporting, on page 551* in the *User Guide*.



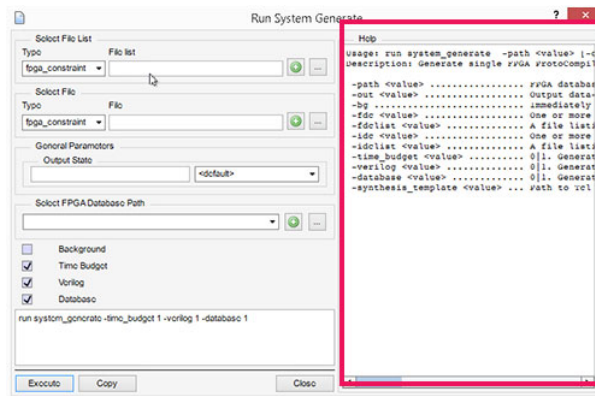
# Task Dialog Boxes

Task dialog boxes are the graphical equivalents of run commands and support a subset of the Tcl command options. Dialog boxes are enabled according to the current database state of the design and displayed by clicking their corresponding entries in the State drop-down menu (see [State Menu, on page 36](#)) or by clicking the corresponding button at the bottom of the State Tree view (see [State Tree Bottom Menu, on page 90](#)). In the ensuing sections, each dialog box is described with its prerequisite condition or database state.

The following dialog boxes are described:

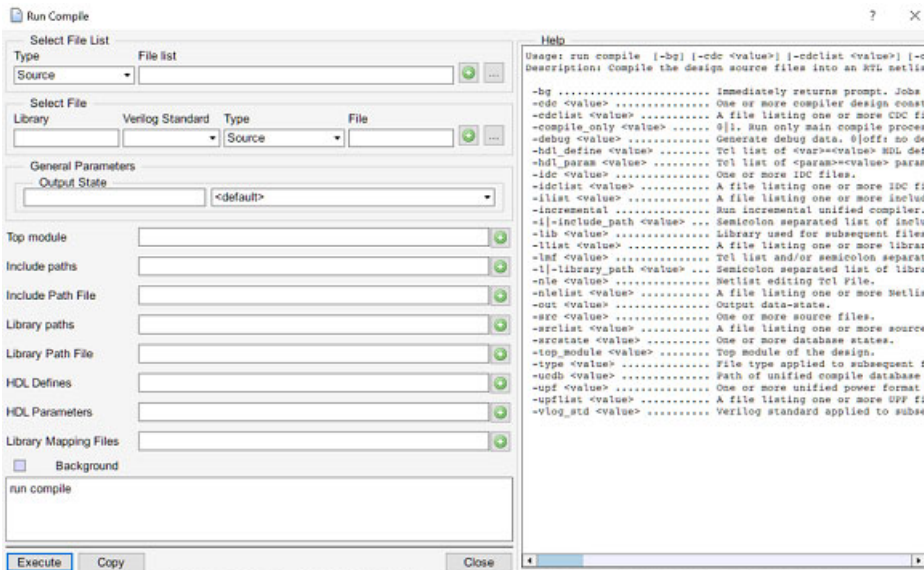
- [Run Compile Dialog Box](#), on page 101
- [Run Instrumentation Preparation Dialog Box](#), on page 103
- [Run Pre-Map Dialog Box](#), on page 105
- [Run Map Dialog Box](#), on page 107
- [Run Pre-Partition Dialog Box](#), on page 108
- [Run Partition Dialog Box](#), on page 109
- [Run System Route Dialog Box](#), on page 111
- [Run System Generate Dialog Box](#), on page 113

Each of the dialog boxes includes a right-sided help panel which lists the complete command syntax (not shown in the individual dialog boxes).



## Run Compile Dialog Box

The Run Compile dialog box is displayed by selecting Run compile from the State drop-down menu on the top-level menu bar. The selection is enabled from root or from the pre-instrument state. Executing the compile process translates the design to the target technology. For additional information, see [Compiling the Design, on page 77](#) in the *User Guide* and [run compile, on page 132](#) in the *Command Reference*.



### Dialog Box Entry

### Function

#### Select File List

##### Type

Drop-down menu that selects the source file input; the available options are source (the default), compiler\_constraint, and ident\_constraint.

##### File list

The path to the file-list file. Use the ... button to navigate to the location of the file list. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.

#### Select File

##### Library

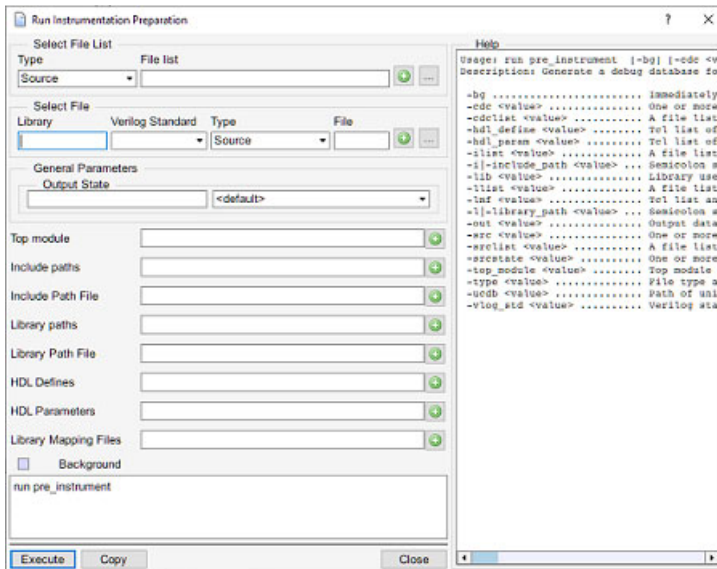
Specifies the Verilog library directory for the selected file.

<b>Dialog Box Entry</b>	<b>Function</b>
Verilog Standard	Drop-down menu that selects the Verilog standard from Verilog 1995, Verilog 2001, or SystemVerilog.
Type	Drop-down menu that specifies the source file type; available values are source (HDL), compiler_constraint, ident_constraint.
File	The selected file. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
<b>General Parameters</b>	
Output State	Allows an output state name to be expressly specified for the compile database state. The <default> drop-down menu lists existing output state names that can be overridden.
Top module	Specifies the name of the top module for the design.
Include paths	Defines the search path used for 'include commands relative to the active directory. Use the Include paths file browse button to locate the path.
Include Path File	A file listing one or more include paths.
Library paths	Defines the search path used for libraries relative to the active directory. Use the Library paths file browse button to locate the path.
Library Path File	Specifies the list of library paths, relative to current working directory, separated with semicolon.
Library mapping files	Specifies a Tcl list of library mapping files.
HDL defines	Specifies a Tcl list of parameters to be applied to the top-level module.
HDL parameters	Specifies a Tcl list of parameters to be applied to the top-level module.
Library mapping files	Specifies a list of library mapping files (lmf) separated with semicolon or in a Tcl file.

Dialog Box Entry	Function
Background	Starts the compilation process and immediately returns the shell prompt. With this setting, license-based jobs are still run sequentially with the next job starting only on completion of the previous job. Unlicensed or third-party jobs are run concurrently.
Status Pane	Reports the dialog box settings to be applied by the run command.
Execute Button	Executes the run compile command with the settings specified and closes the Run Compile dialog.

## Run Instrumentation Preparation Dialog Box

The Run Instrumentation Preparation dialog box is displayed by selecting Run pre-instrument from the State drop-down menu on the top-level menu bar. The selection is enabled only from root. For additional information, see [Instrumenting the Design Before Compiling](#), on page 642 in the *User Guide* and [run\\_pre\\_instrument](#), on page 149 in the *Command Reference*.



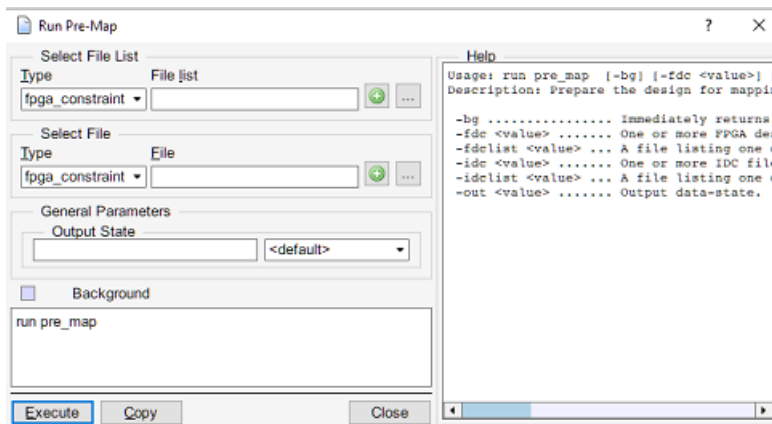
Dialog Box Entry	Function
Select File List	
Type	Drop-down menu that selects the source file input; the available options are source (the default), compiler_constraint, and ident_constraint.
File list	The path to the file-list file. Use the ... button to navigate to the location of the file list. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Select File	
Library	Specifies the Verilog library directory for the selected file.
Verilog Standard	Drop-down menu that selects the Verilog standard from Verilog 1995, Verilog 2001, or SystemVerilog.
Type	Drop-down menu that specifies the source file type; available values are source (HDL), compiler_constraint, or ident_constraint.
File	The selected file. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
General Parameters	
Output State	Allows an output state name to be expressly specified for the compile database state. The <default> drop-down menu lists existing output state names that can be overridden.
Top module	Specifies the name of the top module for the design.
Include paths	Defines the search path used for 'include commands relative to the active directory. Use the Include paths file browse button to locate the path.
Include Path File	A file listing one or more include paths.
Library paths	Defines the search path used for libraries relative to the active directory. Use the Library paths file browse button to locate the path.
Library Path File	A file listing one or more library include paths.



Dialog Box Entry	Function
HDL Defines	Specifies a Tcl list of parameters to be applied to the top-level module.
HDL Parameters	Specifies a Tcl list of parameters to be applied to the top-level module.
Library Mapping Files	Specifies a Tcl list of library mapping files.
Background	Starts the instrumentation preparation process and immediately returns the shell prompt. With this setting, license-based jobs are still run sequentially with the next job starting only on completion of the previous job. Unlicensed or third-party jobs are run concurrently.
Status Pane	Reports the dialog box settings to be applied by the run command.
Execute Button	Executes the run pre_instrument command with the settings specified and closes the Run Compile dialog.

## Run Pre-Map Dialog Box

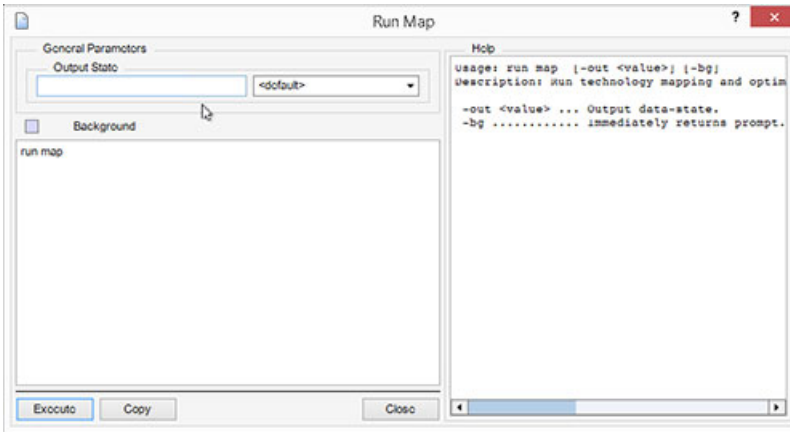
The Run Pre-Map dialog box is displayed by selecting Run pre-map from the State drop-down menu on the top-level menu bar. The selection is enabled only for the synthesis design flow and only after successful execution of a compile task. For additional information, see [Running Pre-Map, on page 474](#) in the *User Guide* and [run\\_pre\\_map, on page 153](#) in the *Command Reference*.



Dialog Box Entry	Function
Select File List	
Type	Identifies the source file input; the available options are <code>fpga_constraint</code> (the default), <code>upf</code> , and <code>ident_constraint</code> .
File list	The path to the file-list file. Use the ... button to navigate to the location of the file list. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Select File	
Type	Specifies an <code>fpga_constraint</code> (the default), <code>upf</code> or <code>ident_constraint</code> file type.
File	The selected file. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
General Parameters	
Output State	Allows an output state name to be expressly specified for the pre-map database state. The <default> drop-down menu lists existing output state names that can be overridden.
Background	Starts the compilation process and immediately returns the shell prompt. With this setting, license-based jobs are still run sequentially with the next job starting only on completion of the previous job. Unlicensed or third-party jobs are run concurrently.
Status Pane	Reports the dialog box settings to be applied by the run command.
Execute Button	Executes the run compile command with the settings specified and closes the Run Pre-Map dialog.

## Run Map Dialog Box

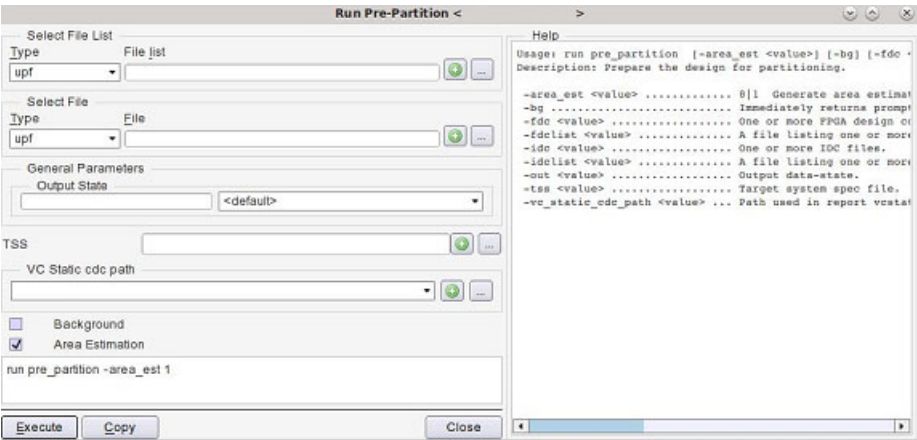
The Run Map dialog box is displayed by selecting Run map from the State drop-down menu on the top-level menu bar. The selection is enabled only for the synthesis design flow and only after successful execution of a pre-map task. For additional information, see [Mapping the Design, on page 490](#) in the *User Guide* and [run map, on page 142](#) in the *Command Reference*.



Dialog Box Entry	Function
General Parameters	
Output State	Allows an output state name to be expressly specified for the map database state. The <default> drop-down menu lists existing output state names that can be overridden.
Background	Starts the mapping process and immediately returns the shell prompt. With this setting, license-based jobs are still run sequentially with the next job starting only on completion of the previous job. Unlicensed or third-party jobs are run concurrently.
Status Pane	Reports the dialog box settings to be applied by the run map command.
Execute Button	Executes the run map command with the settings specified and closes the Run Map dialog.

# Run Pre-Partition Dialog Box

The Run Pre-Partition dialog box is displayed by selecting Run pre-partition from the State drop-down menu on the top-level menu bar. The selection is enabled only for the partition design flow and only after successful execution of a compile task. For additional information, see [Using run pre\\_partition to Define Partitions, on page 324](#) in the *User Guide* and [run pre\\_partition, on page 155](#) in the *Command Reference*.

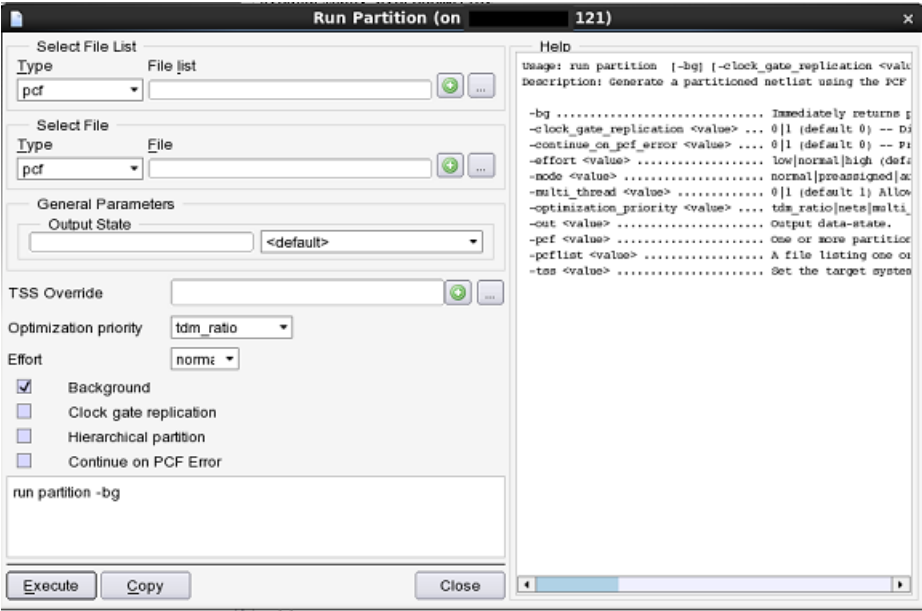


Dialog Box Entry	Function
Select File List	
Type	Identifies the file-list input; the available options are upf (the default), fpga_constraint, and ident_constraint.
File list	The path to the file-list file. Use the ... button to navigate to the location of the file list. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Select File	
Type	Specifies an individual upf, fpga_constraint, or ident_constraint file type.
File	The selected file. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
General Parameters	

Dialog Box Entry	Function
Output State	Allows an output state name to be expressly specified for the pre-map database state. The <default> drop-down menu lists existing output state names that can be overridden.
TSS	Specifies the target physical system specification. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Background	Starts the pre-partition process and immediately returns the shell prompt. With this setting, license-based jobs are still run sequentially with the next job starting only on completion of the previous job. Unlicensed or third-party jobs are run concurrently.
Area Estimation	Generate area estimation data.
Status Pane	Reports the dialog box settings to be applied by the run pre_partition command.
Execute Button	Executes the run pre_partition command with the settings specified and closes the Run Pre-Partition dialog.

## Run Partition Dialog Box

The Run Partition dialog box is displayed by selecting Run partition from the State drop-down menu on the top-level menu bar. The button is enabled only for the partition design flow and only after successful execution of a pre-partition task. For additional information, see [Partitioning the Logic, on page 328](#) in the *User Guide* and [run partition, on page 144](#) in the *Command Reference*.

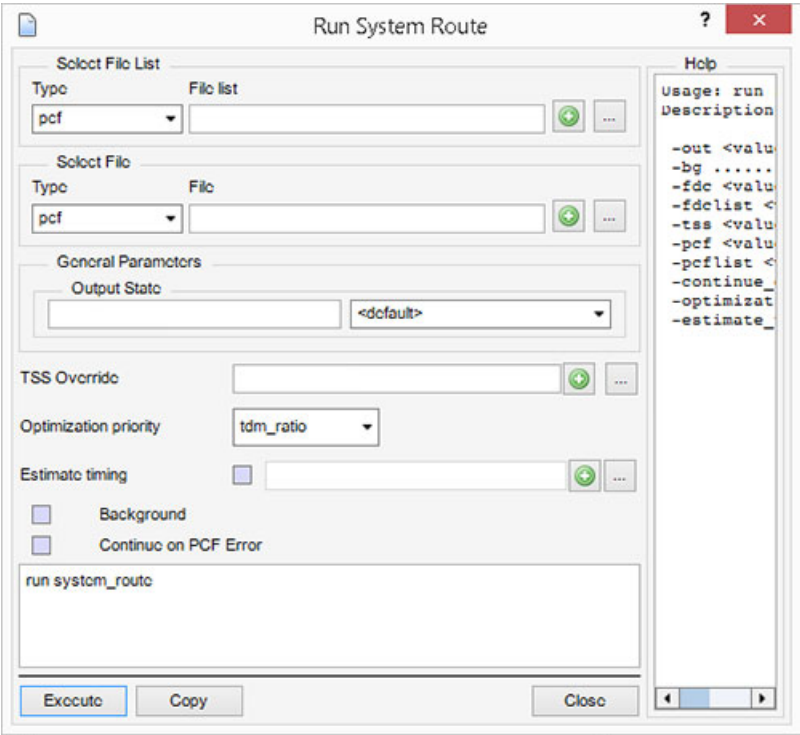


Dialog Box Entry	Function
Select File List	
Type	Identifies the file-list input; the available options are pcf (the default) and fpga_constraint.
File	The path to the file-list file. Use the ... button to navigate to the location of the file list. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Select File	
Type	Specifies an individual pcf or fpga_constraint constraint file type.
File	The selected file. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
General Parameters	

Dialog Box Entry	Function
Output State	Allows an output state name to be expressly specified for the pre-partition database state. The <default> drop-down menu lists existing output state names that can be overridden.
TSS Override	Overrides the target physical system specification from the previous run <code>pre_partition</code> command. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Optimization priority	Sets the objective for the system router.
Effort	Sets the effort level for the partitioner from the adjacent drop-down menu.
Background	Starts the pre-partition process and immediately returns the shell prompt. With this setting, license-based jobs are still run sequentially with the next job starting only on completion of the previous job. Unlicensed or third-party jobs are run concurrently.
Clock gate replication	Automatically replicates clock trees and requisite gates in multi-FPGA designs to eliminate the need for manual clock replication.
Hierarchical partition	Controls the hierarchical partitioning on groups of FPGAs.
Continue on PCF Error	Prints only a warning and continues if the object in the PCF file is not present in the design.
Status Pane	Reports the dialog box settings to be applied by the run partition command.
Execute Button	Executes the run partition command with the settings specified and closes the Run Partition dialog.

## Run System Route Dialog Box

The Run System Route dialog box is displayed by selecting Run system-route from the State drop-down menu on the top-level menu bar. The selection is enabled only for the partition design flow and only after successful execution of a partition task. For additional information, see [Running System Route for the Top Level, on page 412](#) in the *User Guide* and [run\\_system\\_route, on page 160](#) in the *Command Reference*.



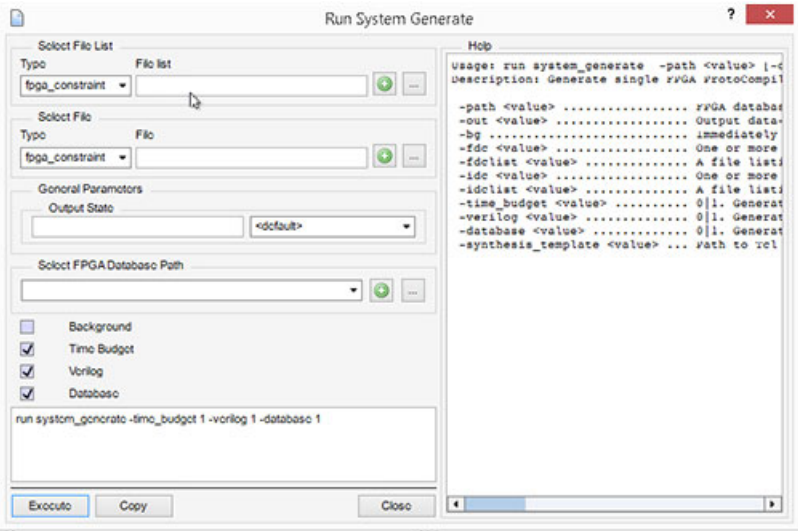
Dialog Box Entry	Function
Select File List	
Type	Identifies the file-list input; the available options are pcf (the default) and fpga_constraint.
File	The path to the file-list file. Use the ... button to navigate to the location of the file list. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Select File	
Type	Specifies an individual pcf or fpga_constraint file type.
File	The selected file. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.



Dialog Box Entry	Function
General Parameters	
Output State	Allows an output state name to be expressly specified for the system-route database state. The <default> drop-down menu lists existing output state names that can be overridden.
TSS Override	Overrides the target physical system specification from the previous run pre_partition command. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Background	Starts the pre-partition process and immediately returns the shell prompt. With this setting, license-based jobs are still run sequentially with the next job starting only on completion of the previous job. Unlicensed or third-party jobs are run concurrently.
Continue on PCF Error	Print a warning and continue if object in pcf file is missing in design.
Effort	Sets the effort level for the partitioner.
Status Pane	Reports the dialog box settings to be applied by the run system route command.
Execute Button	Executes the run system route command with the settings specified and closes the Run System Route dialog.

## Run System Generate Dialog Box

The Run System Generate dialog box is displayed by selecting Run system-generate from the State drop-down menu on the top-level menu bar. The selection is enabled only for the partition design flow and only after successful execution of a system route task. For additional information, see [Generating FPGAs, on page 416](#) in the *User Guide* and [run system\\_generate, on page 157](#) in the *Command Reference*.



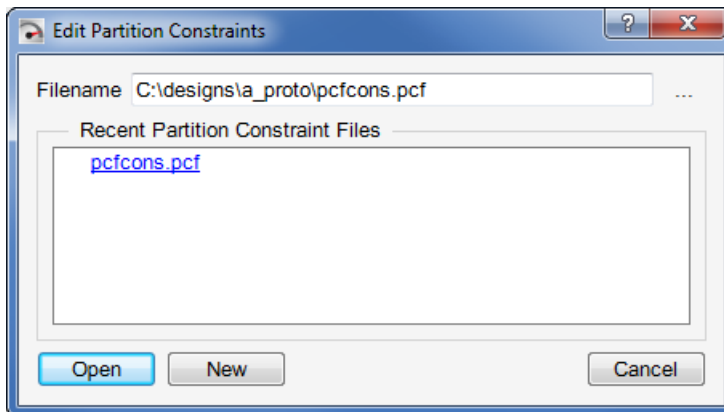
Dialog Box Entry	Function
Select File List	
Type	Identifies the file-list input; the only available option is fpga_constraint.
File list	The path to the file-list file. Use the ... button to navigate to the location of the file list. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
Select File	
Type (for Select File)	Specifies an individual fpga_constraint file type.
File	The selected file. Use the ... button to navigate to the location of the file. The adjacent Add to TCL command icon writes the equivalent Tcl command argument to the lower display area.
General Parameters	
Output State	Allows an output state name to be expressly specified for the system-route database state. The <default> drop-down menu lists existing output state names that can be overridden.
Select FPGA Database Path	Specifies the FPGA database path for synthesis

Dialog Box Entry	Function
Background	Starts the pre-partition process and immediately returns the shell prompt. With this setting, license-based jobs are still run sequentially with the next job starting only on completion of the previous job. Unlicensed or third-party jobs are run concurrently.
Time Budget	Generates time budgets for single FPGA projects.
Verilog	Generates Verilog for single FPGA projects.
Database	Generates SRS files for single FPGA projects.
Status Pane	Reports the dialog box settings to be applied by the run system generate command.
Execute Button	Executes the run system generate command with the settings specified and closes the Run System Generate dialog.

## PCF Graphical Editor

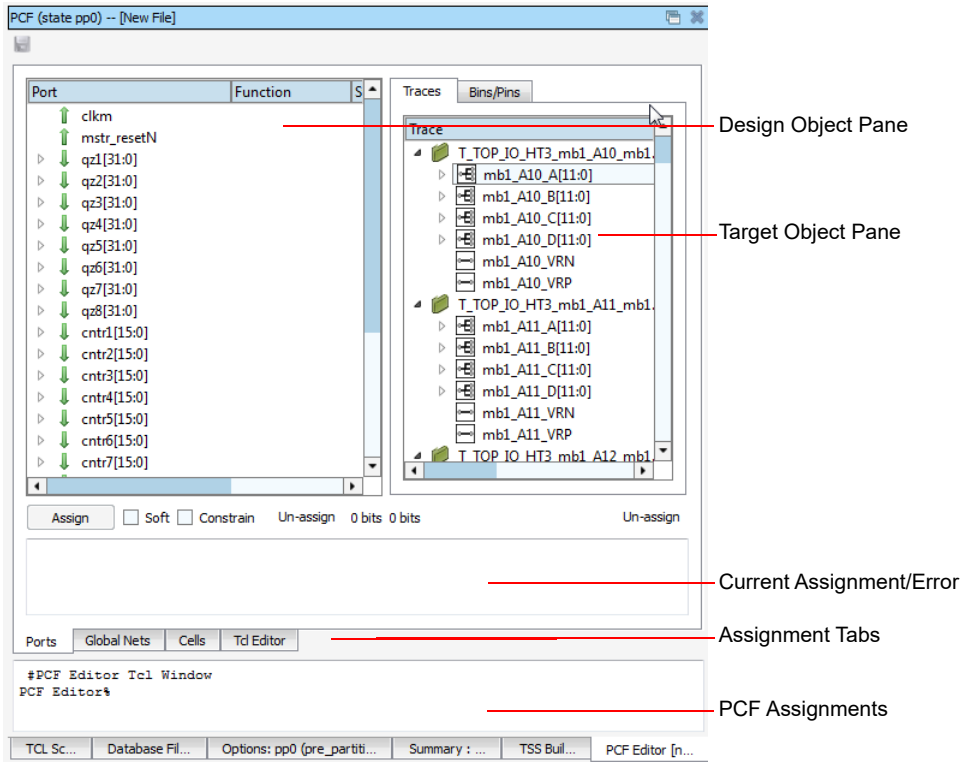
The PCF (Partition Constraint File) editor is an interactive editor for PCF files (see [Chapter 3, \*Partition Constraint File Tcl Commands\*](#) in the Command Reference and [Specifying PCF Constraints in the PCF Editor \(GUI\)](#), on [page 254](#) in the *ProtoCompiler User Guide*). To use the PCF editor, click the Edit Partition Constraints icon or select Edit Partition Constraints from the Tools drop-down menu from an active pre-partition, partition, or system-route database state.

When prompted, either select a file from the filenames displayed or navigate to an existing file using the Filename browser; click either Open or New as appropriate.



## PCF Graphical Editor Window

Clicking the Open or New button opens the PCF graphical editor window.



The PCF graphical editor window can be divided into upper and lower halves:

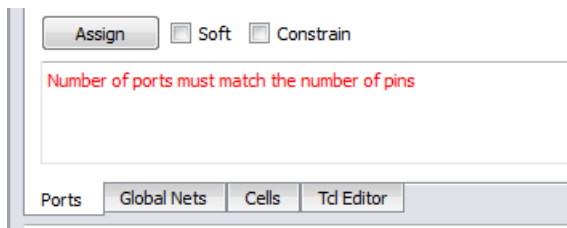
- The upper half consists of these related elements, which are described in detail in the [Ports Tab, on page 119](#), [Global Nets Tab, on page 121](#), [Cells Tab, on page 123](#), and [Tcl Editor Tab, on page 126](#) sections:
  - A design objects pane on the left
  - A target objects pane on the right.
  - A set of tabs (Ports, Global Nets, Cells, and Tcl Editor) that determines the contents of the two upper panes. See the corresponding sections for details.
  - A pane that echoes the PCF command for the current assignment or displays an error message ([Current PCF Constraint, on page 118](#))

- The lower half consists of a Tcl window that reflects all the actions taken and the pcf commands generated by the assignments made in the editor ([PCF Editor Tcl Window, on page 118](#))

## Current PCF Constraint

With the Ports, Global Nets, and Cells panes, when you click the Assign button, the area immediately below the Assign button displays the status for the PCF constraint assignment. For legal assignments, the area displays the PCF constraint.

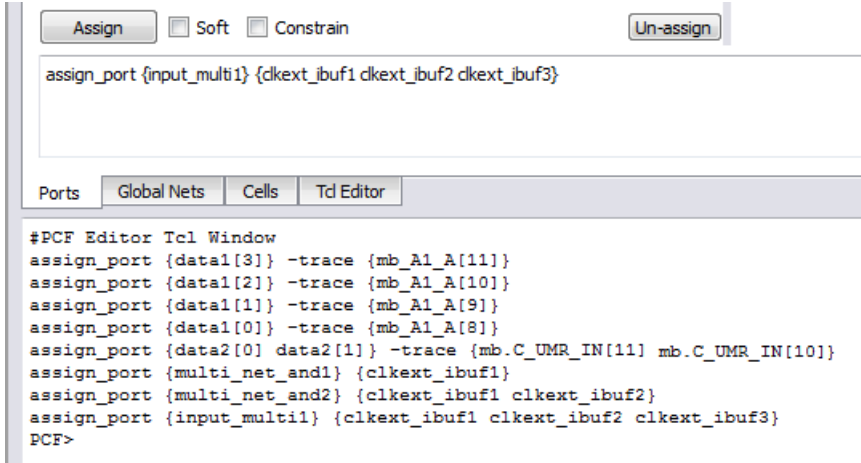
If the attempted assignment is not legal (for example, assigning a bus to a single-bit trace), the pane displays a message in place of the constraint:



The pane does not display assignments that are canceled with Un-assign.

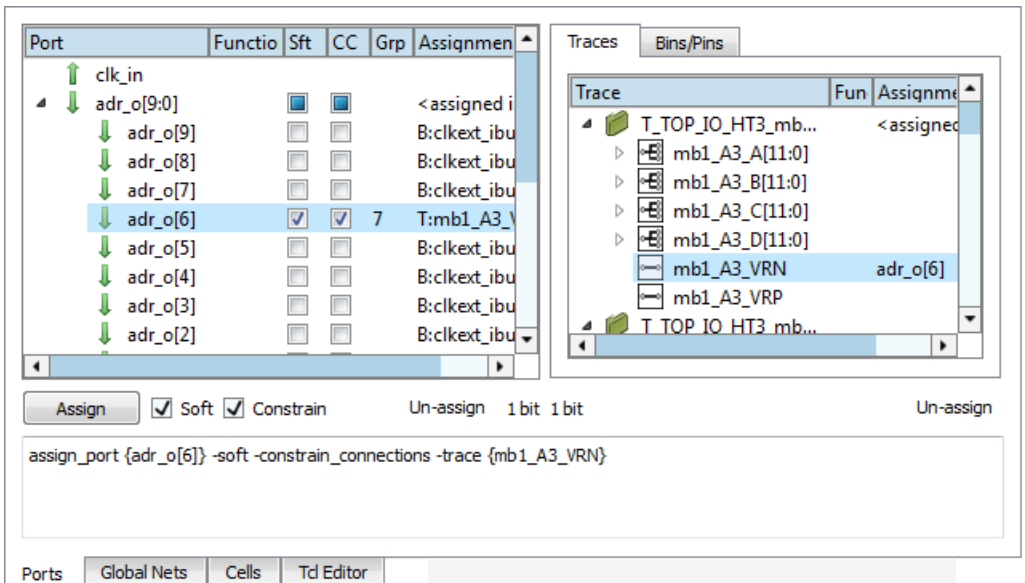
## PCF Editor Tcl Window

As you assign ports to traces or bins/pins, the corresponding commands are echoed in the Tcl Window at the bottom of the PCF Editor window. The window also displays other operations carried out in the window such as messages; it does not display assignments that are canceled as a result of selecting Un-assign.



## Ports Tab

The Ports tab lists the ports in the left section and either the traces or bins/pins in the target section on the right. Ports are assigned by selecting the I/O port in the left section and selecting the corresponding trace or bin in the right section and clicking the Assign button.



## Ports – Design Objects Pane

Port	Lists the ports.
Function	Shows port function set by right-clicking and selecting port attributes from the Net Attributes dialog box (see <a href="#">Net Attributes Dialog Box</a> , on page 121).
Sft	Enabled when the Soft check box (adjacent to the Assign button) is checked to allow the partitioner to ignore constraint limits to improve results for the selected port.
CC	Enabled when the Constrain option is selected. Indicates that the corresponding port cell is restricted to only those bins connected to the specified port or external bins.
Grp	Indicates ports assigned to the same group. Use the Ctrl key to select multiple ports and assign them to the same group.
Assignments	Shows the object from the Target Object pane to which the port is assigned. The assignment is prefixed with a “T” to indicate a trace or a “B” to indicate bin/pin.

## Ports – Target Objects Pane

Traces tab	Lists the traces to which the design objects can be assigned.
Bins/Pins tab	Lists the bins and pins to which the design objects can be assigned.

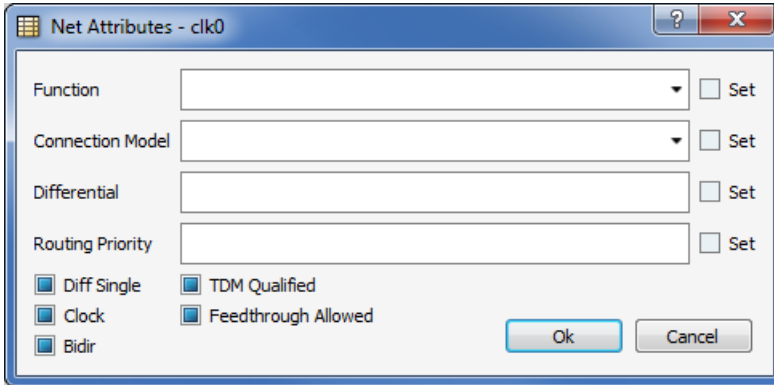
## Ports – PCF Assignment Options

Assign	Assigns the design objects selected in the left pane to the target objects in the right pane and updates the information in both panes accordingly.
Soft	When enabled, indicates that the partitioner can ignore this constraint to improve results.
Constrain	When enabled, indicates that the corresponding port cell is restricted to only those bins connected to the specified port or external bins.
Clear (left)	Clears the current port (and net) assignment; removes the assignment from the display, but does not remove the corresponding command from the PCF Editor Tcl Window.
Clear (right)	Disables the current net (and port) assignment.



## Net Attributes Dialog Box

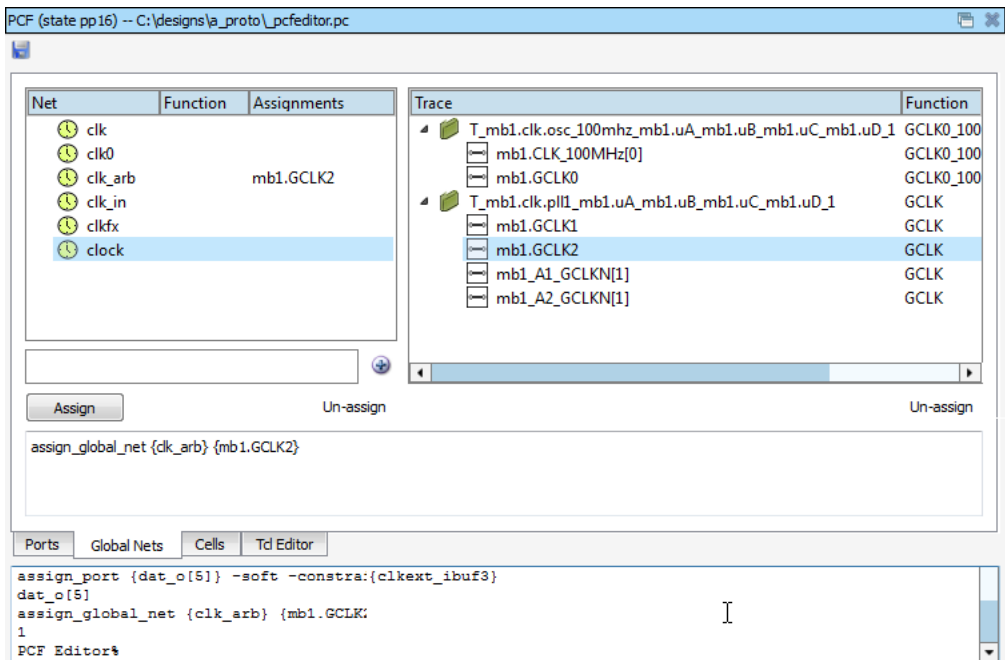
The Net Attributes dialog box assigns parameters to the selected nets or global nets in the left panel.



Function	Selects the port function from a drop-down list; functions include GCLK, GCLK_100MHz, RESET, and five_fixed_traces.
Connection Model	Identifies net as direct (non-multiplexing) or TDM capable.
Differential	Limits <i>netNameList</i> to single nets.
Routing Priority	Defines the routing priority ranging from 1 (highest) to 255.
Diff Single	Route the net on the P trace and preserves the N trace.
Clock	Limits net to traces with clock-capable pins.
Bidir	Limits net to traces with bidirectional capability.
TDM Qualified	Indicates that the listed nets are qualified for time-domain multiplexing.
Feedthrough Allowed	Allows the net to pass through intermediate devices.

## Global Nets Tab

The Global Nets tab lists the system-defined global nets in the left section and the target traces in the right section. Global nets are assigned by selecting the net in the left section and selecting the corresponding trace in the right section and clicking the Assign button. You can drag nets from the HDL analyst schematic view directly on to the Global Nets tab.



## Global Nets – Design Objects Pane

Net	Lists the global nets.
Function	Shows net function set by right-clicking and selecting the net attribute from the Net Attributes dialog box (see <a href="#">Net Attributes Dialog Box</a> , on page 121).
Assignments	Shows the object from the Target Object pane to which the net is assigned.
Type-in Field	Type names of global nets to add to the list. You can also drag and drop nets from an HDL Analyst view.

## Global Nets – Target Objects Pane

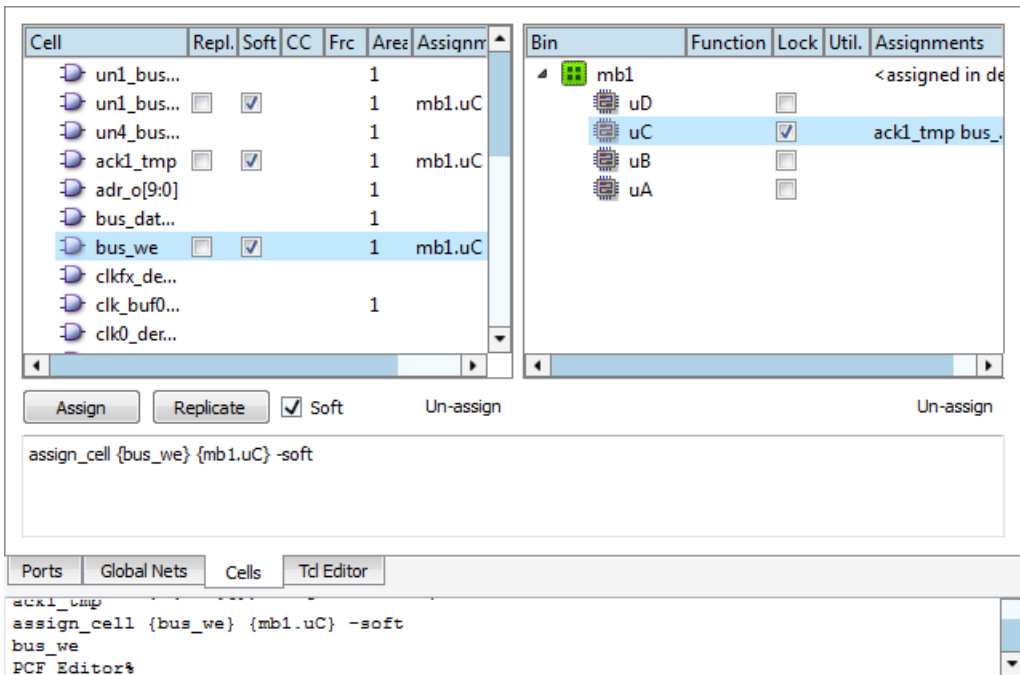
Trace	Lists the traces to which the design objects can be assigned.
Function	Reports the trace function.
Assignments	Shows the assigned design object once an assignment is made.

## Global Nets – PCF Assignment Options

### Cells Tab

Assign	Assigns the design objects selected in the left pane to the target objects in the right pane and updates the information in both panes accordingly.
Un-assign	Clears the current net (and trace) assignment; removes the assignment from the display, but does not remove the corresponding command from the PCF Editor Tcl Window.
Un-assign	Clears the current trace (and net) assignment.

The Cells tab lists the cell instances in the left section and the target FPGA bins in the right section. Cells are assigned or replicated by selecting the cell in the left section and selecting the corresponding FPGA bin or bins in the right section and clicking the Assign or Replicate button.



## Cells – Design Objects Pane

Cell	Lists the cell instances.
Repl	Indicates if the selected cell is to be replicated (the cell must be assigned to a bin).
Soft	Enabled when the Soft check box (adjacent to the Replicate button) is checked to allow the partitioner to ignore constraint limits to improve results for the selected cell.
CC	Enabled when the cell is targeted for replication (Repl checked). Checking the CC box restricts the port cell to only those bins connected to the specified port or external bins.
Frc	Enabled when the cell is targeted for replication (Repl checked). Checking the Frc box forces cell replication even when the cell does not include output connections in the target bin.
Area	Indicates the unit area for the cell.
Assignments	Shows the object from the Target Object pane to which the cell is assigned.

## Cells – Target Objects Pane

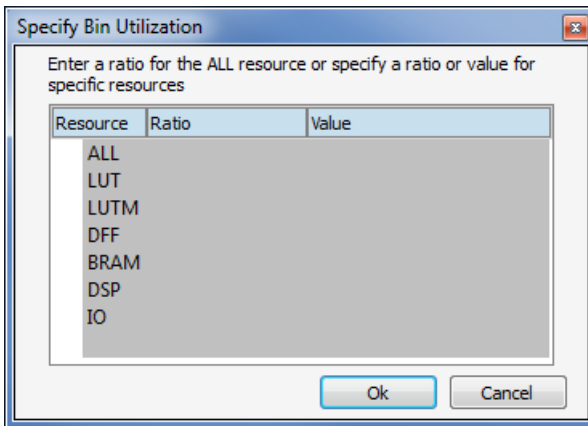
Bin	Lists the target FPGA bins.
Function	Reports function assignments for the listed traces.
Lock	Indicates if the bin is locked.
Util	Specifies the value of the limit for the named resource through the Bin Utilization dialog box (see <a href="#">Bin Utilization Dialog Box</a> , on page 125).
Assignments	Shows the assigned design object once an assignment is made

## Cells – PCF Assignment Options

Assign	Assigns the design objects selected in the left pane to the target objects in the right pane.
Replicate	Forces cell replication even when the cell does not include output connections in the target bin.
Soft	When enabled, indicates that the partitioner can ignore this constraint to improve results.
Un-assign	Clears the current cell (and bin) assignment; removes the assignment from the display, but does not remove the corresponding command from the PCF Editor Tcl window.
Un-assign	Clears the current bin (and cell) assignment.

## Bin Utilization Dialog Box

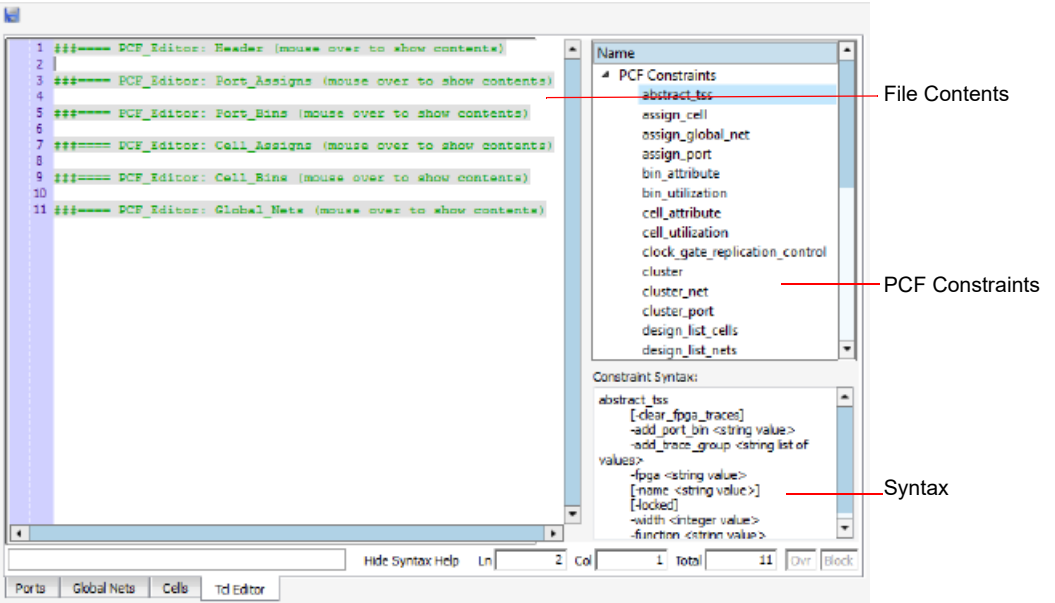
The Specify Bin Utilization dialog box enables changing the resource limits of FPGA bins. Dialog box selection allows only one resource-value pair per entry.



Resource	A list of the applicable cell resources. ALL applies the specified ratio to all resources.
Ratio	Specifies a ratio relative to the default value for the selected FPGA to limit the available resources. Applies to individual resources or ALL resources. The ratio is a floating point value between 0 and 1.
Value	Used with individual resources to specify a value relative to the default bin value.

# Tcl Editor Tab

The primary function of the Tcl Editor window is to show the Tcl command entries to be generated based on the assignments made in the other tabs. The window also lets you add your own entries to the file. The window consists of three main panes: the file contents, a list of the PCF commands available, and the complete command syntax for a selected command.



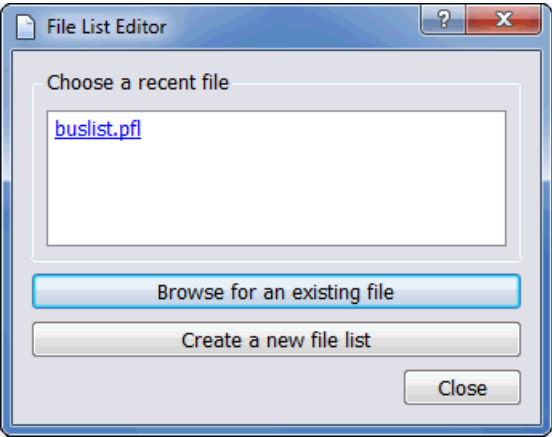
Contents	Shows the different sections of the PCF file. Move the cursor over a heading to see the commands in that section (generated from assignments made on other tabs). To enter a command, type in the command in the appropriate section of the file.
PCF Constraints	Shows available PCF commands. Select a command to display the syntax in the Syntax box. If this is not displayed, click Syntax Help. Double-click an entry to add it to the contents window.

Constraint Syntax	Displays the syntax for the selected PCF constraint. If this is not displayed, click Syntax Help.
Hide Syntax Help Syntax Help	Toggles between displaying and hiding the PCF Constraint and Syntax sections.
Save icon (upper left corner)	Saves the PCF file.

# File List Editor



The File List Editor is invoked by clicking the Edit File List icon in the main menu or by selecting Edit File List from the Tools drop-down menu.

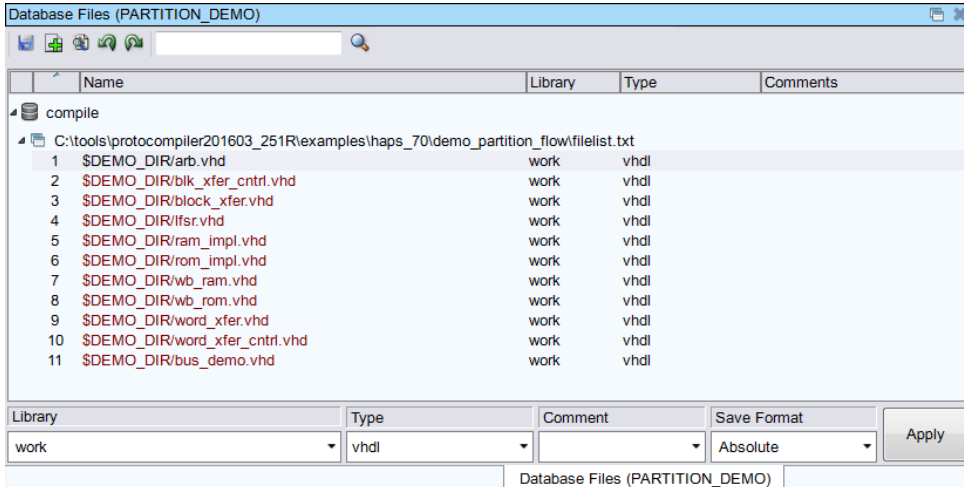


Dialog Box Entry	Function
Choose a recent file	Opens the selected list file in the File List Editor; see <a href="#">File List Editor (Detail)</a> , on page 128.
Browse for an existing file	Opens a file browser to allow you to locate an exiting list file.
Create a new file list	Opens a file browser to allow you to specify a directory location for the file list.

## File List Editor (Detail)

Choosing a file from the initial File List Editor dialog opens the detailed dialog box for creating or modifying a file-list file.



**Dialog Box Entry****Function**

FL-Edit: (top banner)

Lists the full path to the file-list file being edited.

Save icon

Saves the specified file-list file following modification.

Add Files icon

Opens the Select Files dialog box to select files to be added. Selecting a file adds the file to the file list display (see [Select Files Dialog Box](#), on page 131).

Find in Files icon

Opens the Find in Files dialog box to search for strings within the listed files (see [Find in Files Dialog Box](#), on page 130).

File Search Field

Lists only the names of files that include the string entered in the search field; clicking the adjacent search icon initiates the search, and clicking the 'x' redisplay the complete file list.

File display area

Lists the files in the file list. Each entry includes its file order, associated library and type, and its absolute or relative file name. An optional comments field is also available. Clicking on an entry enables the Library, Type, Comment, and Save Format selections across the bottom of dialog box to modify the corresponding file-list entry. Double clicking on a file name opens the file in a text editor.

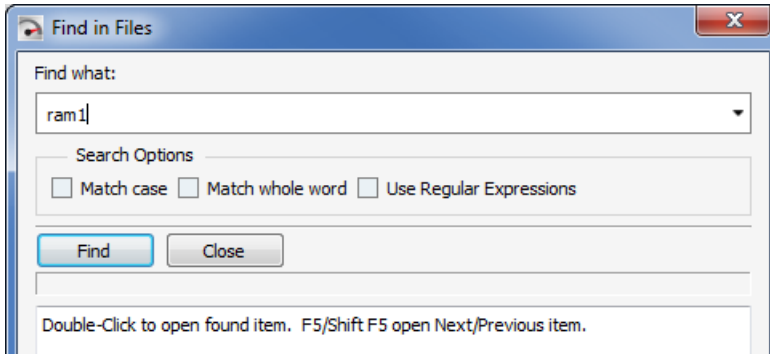
File order column

Displays the order of files in the file list.

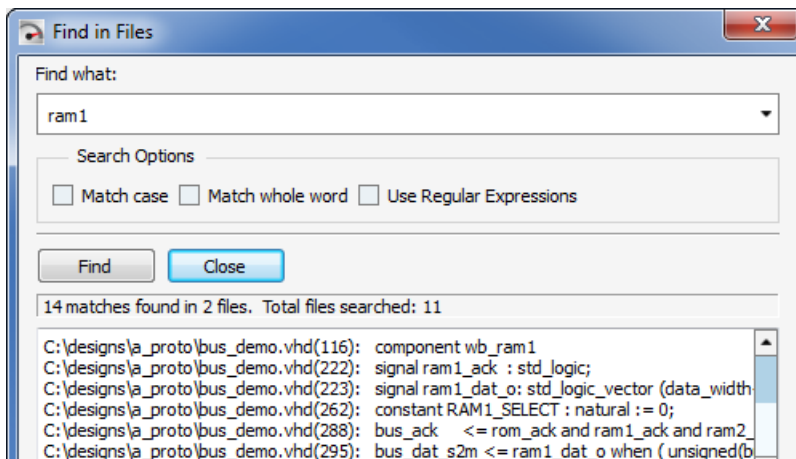
<b>Dialog Box Entry</b>	<b>Function</b>
Library column	Lists the associated library for each file-list entry.
Type column	Lists the associated file type for each file-list entry.
Name column	Lists the name of the file-list entry; an entry can be either absolute or relative.
Library drop-down menu	Defines the library for the selected (highlighted) entry in the file display area. Clicking the Apply button updates the entry in the display.
Type drop-down menu	Defines the file type for the selected (highlighted) entry in the file display area. Clicking the Apply button updates the entry in the display.
Comment field	Allows a comment to be added to the selected (highlighted) entry in the file display area. Clicking the Apply button adds the comment to the display.
Save Format drop-down menu	Determines if paths (absolute) are included with the name entry in the file display area. Use caution when using relative paths as they are relative to the working directory of the corresponding run command.
Apply button	Updates any changes to the entries in the file display area (changes are not saved until the file is actually saved). Changes can be reverted (before the file is saved) using the Undo/Redo icons at the top of the dialog box.

## Find in Files Dialog Box

The Find in Files dialog box searches for text strings within the listed files.

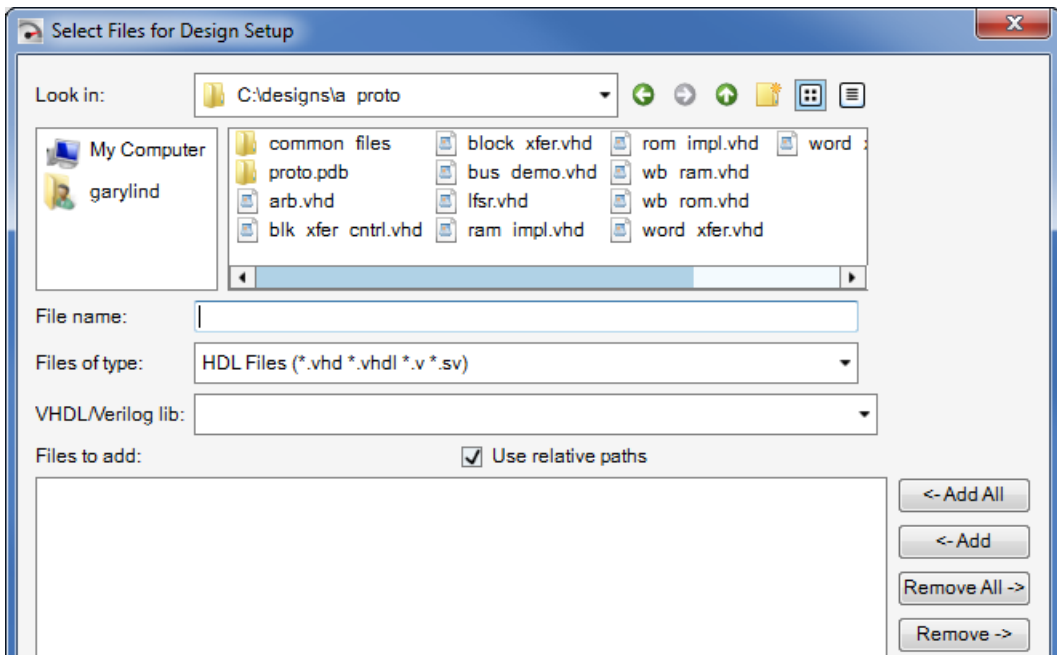


Entering a string in the Find what: field and clicking the Find button, lists the line number of each match found and a context. The files are searched in listed order, and the number of matches found, the number of files with matches, and the total number of files searched is reported. Double clicking on a search result opens the corresponding file in a text editor. Pressing F5 opens the file with the next occurrence; if it is in the same file, the cursor advances to the next occurrence of the string. Pressing SHIFT + F5 opens the file with the previous occurrence; if it is in the same file, the cursor moves back to the previous occurrence of the string.



## Select Files Dialog Box

The Select Files dialog box is displayed by selecting the Add Files icon.



Dialog Box Entry	Function
Look in: field	Selects the path to the files.
File Display	Lists the files in the directory path.
Files of type: field	Filters the file types displayed from the drop-down list of file types.
VHDL/Verilog lib: field	Drop-down menus for library, Verilog standard, and file type.
Files to add: field	Lists selected files.
Use Relative Paths check box	Makes file names relative or absolute in the file list. Use caution when using relative paths as they are relative to the working directory of the corresponding run command.
Add All button	Adds all qualified files to the lower display area.

Dialog Box Entry	Function
Add button	Adds the selected files from the file display to the files listed in the lower display.
Remove all button	Removes all files listed in the lower display.
Remove	Removes individually select files listed in the lower display.

# Options Editor



The options editor allows option settings to be changed from a graphical interface. The editor can be opened from any database state by clicking the Edit Options icon in the top menu bar. By default, the settings of all options are listed; selecting a database state from the State drop-down menu lists only the options pertinent to that state.

For more information about working with options, see [Setting Options, on page 171](#) in the *User Guide*.

Options 📄 ✖

🏠 💾 🔄 State: All

Label	Name	Current value
Fanout Guide	maxfan	10000
Multiple File Compilation Unit	multi_file_compilation_unit	<input type="checkbox"/>
Disable Constraint Check	no_constraint_check	<input type="checkbox"/>
Disable Sequential Optimizations	no_sequential_opt	<input type="checkbox"/>
Optimize NGC/EDIF	optimize_ngc	<input checked="" type="checkbox"/>
Pipelining	pipe	<input checked="" type="checkbox"/>
Enable Xilinx Readback Capabilities	prepare_readback	<input type="checkbox"/>
Resolve Mixed Drivers	resolve_multiple_driver	<input checked="" type="checkbox"/>
Resource Sharing	resource_sharing	<input checked="" type="checkbox"/>
Retiming	retiming	<input type="checkbox"/>
Automatic Read/Write Check Insertion for RAM	rw_check_on_ram	<input checked="" type="checkbox"/>
FPGA Speed Grade	speed_grade	-1
Implicit Initial Value Support for Instantiated ...	support_implicit_init_netlist	<input type="checkbox"/>
Implicit Initial Value Support	supporttypedfit	<input type="checkbox"/>
Synthesis Pragma	synthesis_onoff_pragma	<input type="checkbox"/>
Synthesis Strategy	synthesis_strategy	routability

Name: design\_flow  
 Default Value: "partition"  
 Description: Set the ProtoCompiler design flow

■ Option value differs from value when selected state was last run. See tooltip...

Options Editor Entry	Function
State drop-down menu	Limits the display to only options pertinent to the selected state; the default is ALL.
Select Option Tcl File icon	Opens the Select Option Tcl File dialog box to allow selection of the Tcl file for editing
Save Option as Tcl File icon	Displays the Save Option as Tcl File dialog box to allow the edited file to be saved.
Label field	The GUI name of the option.
Name field	The equivalent name in the option set command syntax.
Current value field	Lists the current option value; the value is highlighted (in yellow) if it differs from the original option setting when the state was last run.
Description field	Brief description of the option and its default value.

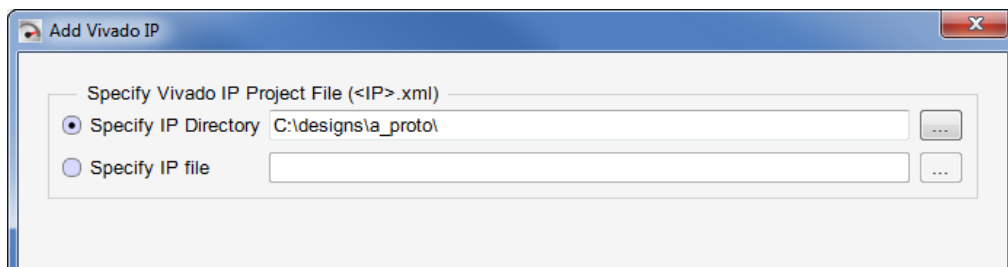
## Add Vivado IP

The first page of the Add Vivado IP dialog box is displayed by clicking the Import Vivado IP entry in the Tools drop-down menu. The IP generated by Vivado IP Catalog consists of a directory that includes a set of files. Each IP directory contains three primary files: *ip.xci*, *ip.xml*, and *ip.dcp*. ProtoCompiler reads these files during IP import.

---

**Note:** To successfully add a Vivado IP, you either need both *ip.xci* and *ip.xml* files, or an *ip.dcp* file.

---



- **Specify IP file** – To import a single IP block, enter any of the *ip.xci*, *ip.xml*, or *ip.dcp* files in the Specify IP file field. If you enter either *ip.xci* or *ip.xml* file, ProtoCompiler searches the availability of the other file. For example, if you specify the *ip.xci* file, ProtoCompiler searches for the *ip.xml* file. Vivado IP is added successfully if the other file is found. To import more than one IP block with a single operation, create a file list of the *ip.xci*, *ip.xml*, and *ip.dcp* files to be imported and enter the name of the file list in the Specify IP file field.
- **Specify IP Directory** – To import multiple IP blocks, specify the top-level IP directory in the Specify IP Directory field.

The second page of the dialog box selects the IP blocks to be imported, and the third page specifies the synthesis mode and the location for the imported IP. See [Importing Vivado IP, on page 344](#) in the *ProtoCompiler Compiler and Mapper Guide* for additional information.

A Tcl command is also provided to duplicate the dialog box functions; see [generate import\\_vivado\\_ip, on page 64](#) in the *ProtoCompiler Command Reference* for the complete syntax.



# Verdi Debug Setup

The Verdi software is integrated with the prototyping software to provide a seamless design debug environment to validate designs.

You can launch Verdi Debug only from map state or system\_generate state.

The Verdi Debug Setup dialog box is displayed by selecting Verdi Debug from the State drop-down menu.

Verdi Debug Setup Field	Description
-------------------------	-------------

Required Setup	
Run Directory	Run directory containing the debug data for executing Verdi.
VCS_HOME	Path to the directory containing VCS. Note that the tool version should match the release requirements.

Verdi Debug Setup Field	Description
VERDI_HOME	Path to the directory containing Verdi. Note that the tool version should match the release requirements.
XILINX_VIVADO	Path to the directory containing Xilinx-Vivado. Note that the tool version should match the release requirements.
Miscellaneous	
UC 2.0 RTL simv.daidir (use only when required)	Absolute path to the VCS simv.daidir for UC2.0 flow.
Data Expansion Force Value File	Specifies file containing forced values for Data Expansion.
Data Expansion RTL Signal File (Beta)	(Beta) Specifies the file containing RTL hierarchical signal paths for Data Expansion
Only Generate Correlation Database	Select the check-box to only generate correlation database.
Fast Correlation Flow using Verdi “AdvancedHAPSDebug” license (Beta)	(Beta) Select this option to use the enhanced correlation flow. This flow reduces the runtime using Verdi <i>AdvancedHAPSDebug</i> license.
FSDB	
FSDB Files	Absolute path to the FSDB file.
Top Module	Specifies the FSDB top module name.
Type (input format used for PNR run)	Specifies the FSDB format type to be used for place and route run. Supported types are <code>edif</code> , and <code>verilog</code> . Default value is <code>edif</code> . To generate correlation database, the FSDB type must be specified. But other options are not required.

Verdi Debug Setup Field	Description
Only Run Data Expansion	Select this option to only run Data Expansion using an existing correlation database.
Export UC2.0 VCS RTL simv.daidir	Select the option to export the UC 2.0 VCS RTL simv.daidir.
Create Links	Select this option to create links. Creating links saves disk space, but requires access from the exported location.

See [Integrated Debugging with the Verdi Software](#) in the HAPS Prototyping User Guide for additional information.

A Tcl command is also provided to duplicate the dialog box functions; see [launch verdi](#) in the *ProtoCompiler Command Reference* for the complete syntax.



## CHAPTER 4

# Feature Descriptions

---

This chapter includes background and other information for some features included in the prototyping tool:

- [Unified Compile Support](#), on page 142
- [UPF Support](#), on page 144
- [Time Budgeting](#), on page 146

# Unified Compile Support

Unified Compile (UC) uses the VCS<sup>®</sup> software instead of the native compiler, to parse and elaborate the design and generate a database. Using a common front-end allows the same design to be easily migrated from one Synopsys<sup>®</sup> verification tool to another, from simulation to prototyping.

For more information about UC, refer to these descriptions in the *User Guide*:

- [Using Unified Compile](#), on page 120
- [Converting Designs for Unified Compile](#), on page 920
- [Handling HDL Differences](#), on page 921

For details about the VCS compiler, refer to the VCS documentation.

## Compiler Comparison

The UC flow is an alternative starting point to the native compiler built into the prototyping tool, and the following table compares features in the two compilers. For information about using the native compiler, see *Creating and Compiling Databases > Compiling the Design* in the *HAPS<sup>®</sup> Prototyping User Guide*.

Native Compile	Unified Compile
Uses top-level module and Verilog standard specified by the -top_module and -vlog_std arguments respectively.	Uses the top-level module and Verilog standard defined in the Unified Compile database (ucdb).
Supports UUM (Library Mapping Files) to match the VCS software when targeting HAPS. Sets up the software for UUM using LMF.	Re-uses the existing VCS compilation scripts to target HAPS.
Native front-end compiler analyzes and elaborates the RTL files.	The VCS tool analyzes and elaborates RTL files and generates a word-level netlist.
Supports UPF 1.0 and 2.0 constructs defined in UPF file. Supports the standard Tcl syntax.	UPF is processed by VCS (unified UPF).
Supports predefined macros and compilation directives in RTL.	Use the VCS command line options to define all synthesis macros and compiler directives.

<b>Native Compile</b>	<b>Unified Compile</b>
User-Defined Primitives (UDP) are not supported.	UDP are supported.
EDIF/NGC input supported	EDIF/NGC input is supported as input to run compile along with the ucdb option; not supported in the VCS script.
P1735 and Synopsys encryption (synenc) standards are supported.	Supported if encryption is handled by the VCS tool. (P1735 and synenc with the VCS key).
Source-level partitioning flow support	Source-level partitioning not supported.
Monolithic/distributed compiler mode supported.	Only distributed compiler mode supported.

# UPF Support

The ProtoCompiler and ProtoCompiler DX products facilitate the porting of ASIC designs to an FPGA by supporting a subset of the Unified Power Format (UPF) standard. Power-aware FPGA synthesis is based on virtual power domains and, by its nature, does not support commands or options that control power/ground nets and power switches. The supported FPGA UPF commands are related to isolation, retention, power domains, and scope resolution.

The UPF information is specified differently, depending on whether UC or the standard compiler is used for the compile stage.

For more information about the using UPF with the native compiler, see the following topics:

- [Including UPF Specifications](#), on page 827 in the *User Guide*
- [Chapter 5, UPF Commands](#) in the *Command Reference*

For information about the using UPF with the UC flow, see these topics:

- [Setting up the UTF File and the VCS Script](#), on page 128 in the *User Guide*
- [UPF File for Unified Compile](#), on page 188 in the *Reference Manual*

## UPF Features

The tools support isolation, retention, and power domain definitions:

- Power domains can be specified either from the top down or from the bottom up. You can also specify a power domain that is made up of a collection of specific cells.
- Isolation is the strategy that separates design elements without power from parts of the design that are still active and have power. Unexpected values can result when the output of powered-off modules drives active, powered-on modules which can cause incorrect logic behavior. The UPF standard allows for the insertion of isolation cells between power domains. The cells are powered by a constant supply and “clamp” the output of the powered-off domain to ensure a fixed value during its powered-down state. Specify at least one isolation strategy for every power domain created.



- Retention is the strategy that maintains state values when power is disabled. Normally, when power is turned off, the state value is lost. Some components require that these values be retained to avoid a degradation in performance or to ensure quick restarts when power is reapplied. The UPF standard allows specification of flip-flops for retention.

# Time Budgeting

The software automatically implements timing budgets when it partitions the design and generates individual FPGAs. This section describes some details about how the tool implements `set_datapathonly_delay` constraints in the `fdc` files for each FPGA.

See the following for details:

- [Time Budgeting Between Partitions](#), on page 146
- [Time Budgeting Calculation](#), on page 152
- [Generating FPGAs](#), on page 416 in the *User Guide*
- [set\\_datapathonly\\_delay](#), on page 317 in the *Command Reference*

## Time Budgeting Between Partitions

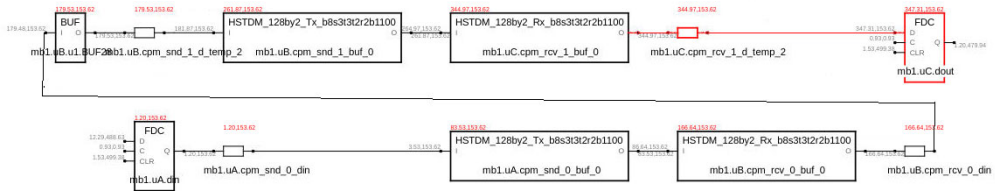
The following examples show how the tool assigns timing budgets between partitions in some cases:

- [Time Budgeting with Multi-Hop Paths](#), on page 146
- [Time Budgeting with Multicycle Constraints](#), on page 147
- [Time Budgeting for Direct Paths with Multiple Clocks](#), on page 148
- [Time Budgeting for Combination Paths \(Multi-Hop and Direct\)](#), on page 150

For information about how the values are derived, see [Time Budgeting Calculation](#), on page 152.

### Time Budgeting with Multi-Hop Paths

The example below shows a multi-hop path from FPGA uA to uC through uB.



The tool creates the following constraints for uB to set the timing budget from receiver to sender:

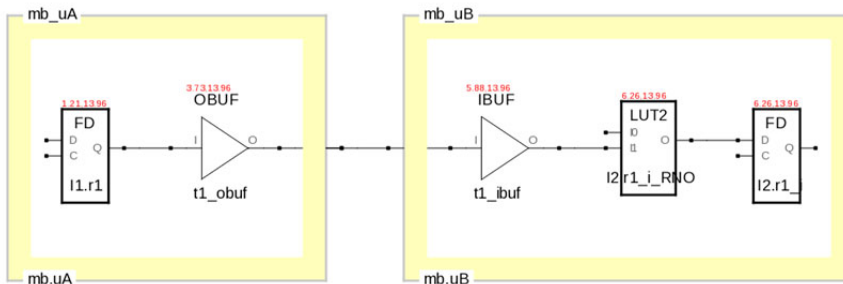
```
set cpm_rcv_0_mserdes__cpm_snd_1_mserdes__0_0 [expr 123.0]

set_datapathonly_delay -comment {HSTDM hop budget
  cpm_rcv_0(mserdes)__cpm_snd_1(mserdes)}
  -from {i:cpm_rcv_0.rxdemux.out.data[*]} -to {i:cpm_snd_1.in_data[*]}
  $cpm_rcv_0_mserdes__cpm_snd_1_mserdes__0_0
```

## Time Budgeting with Multicycle Constraints

The example below shows a multicycle constraint on a path that is split between two FPGAs. The clock period for clk1 is 10 ns.

```
set_multicycle_path -from {i:l1.r1} -to {i:l2.r1} 2
```



For direct paths like this one, the tool sets datapath delays for the timing budget, not input and output delays. The tool adds the following constraints to each FPGA to correctly model the multicycle path exception, based on the available period of 20 ns (2 cycles) set with the multicycle constraint:

mb\_uA\_timing.fdc:

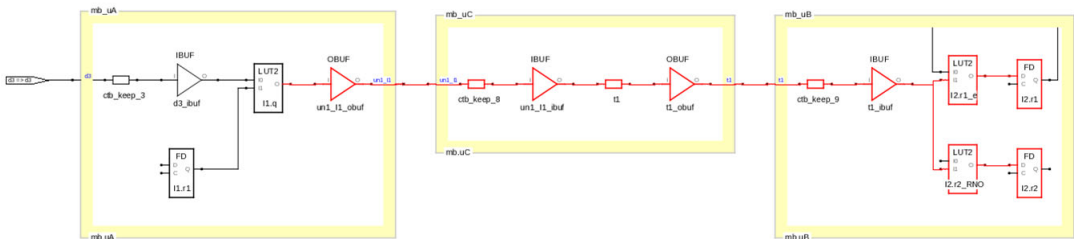
```
# Time budgeting constraints for direct paths
# BudgetableDelay (2.5) + SlackDis (9.6) = 12.1
set datapathonly_delay -rise_from [get_clocks -include_generated_clocks {clk1}]
  -to {p:t1} 12.1
```

mb\_uB\_timing.fdc:

```
# BudgetableDelay (1.2) + SlackDis (4.4) = 5.6
set datapathonly_delay -rise_to [get_clocks -include_generated_clocks {clk1}]
  -to {p:t1} 5.6
```

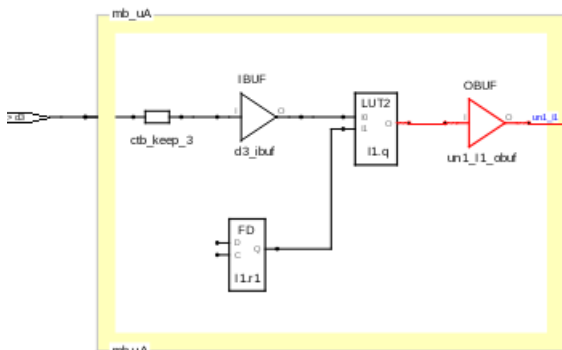
## Time Budgeting for Direct Paths with Multiple Clocks

For direct paths with multiple clocks, the tool sets separate `set_datapathonly_delay` constraints on the I/O ports for each associated clock.



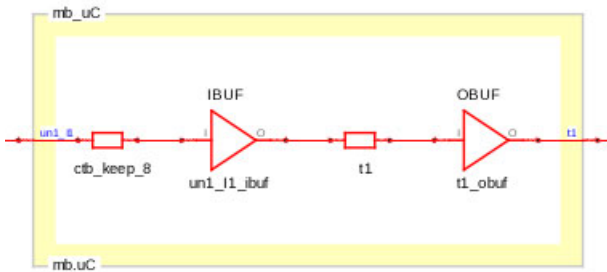
For the path from uA to uC to uB shown in the example above, the tool sets the following constraints for the sender (Tx), intermediate, and receiver (Rx) FPGA:

- mb\_uA (Sender)  
Separate datapath delay constraints for each associated clock



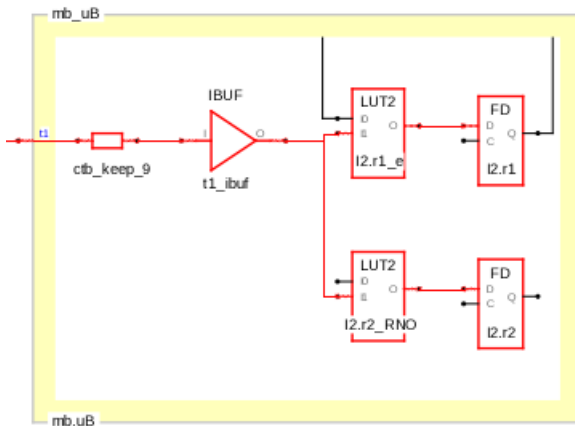
```
# Time budgeting constraints for direct paths
# BudgetableDelay (2.9) + SlackDis (36.1) = 39.0
set datapathonly_delay -rise_from [get_clocks -include_generated_clocks {clk1}]
  -to {p:un1_I1} 39.0
# BudgetableDelay (3.8) + SlackDis (36.5) = 40.2
set datapathonly_delay -rise_from [get_clocks -include_generated_clocks {clk2}]
  -to {p:un1_I1} 40.2
```

- **mb\_uC (Intermediate)**  
DPO (datapath only) constraints from input port to output port



```
# Time budgeting constraints for direct paths
# BudgetableDelay (3.3) + SlackDis (41.1) = 44.4
set datapathonly_delay -from {p:un1_I1} to 39.0 {p:t1} 44.4
```

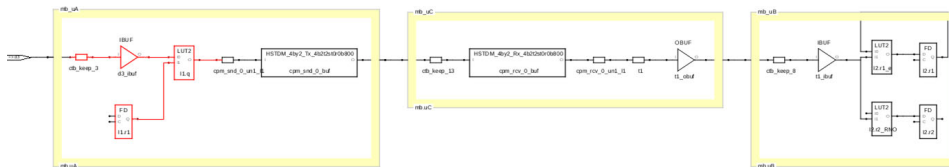
- **mb\_uB (Receiver)**  
Separate port-to-clock DPO constraints for each associated clock



```
# Time budgeting constraints for direct paths
# BudgetableDelay (1.2) + SlackDis (15.1) = 16.3
set datapathonly_delay -rise_to [get_clocks -include_generated_clocks {clk1}]
  -from {p:t1} 16.3
# BudgetableDelay (1.2) + SlackDis (15.1) = 16.3
set datapathonly_delay -rise_to [get_clocks -include_generated_clocks {clk2}]
  -from {p:t1} 16.3
```

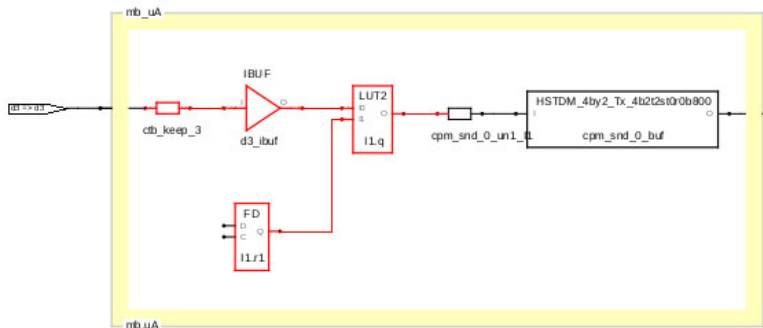
## Time Budgeting for Combination Paths (Multi-Hop and Direct)

The following example shows a combination of direct and HSTDM paths, with partial hops. In cases like this, the timing budget is driven by TDM.



The tool writes out these timing budget constraints for the individual FPGAs:

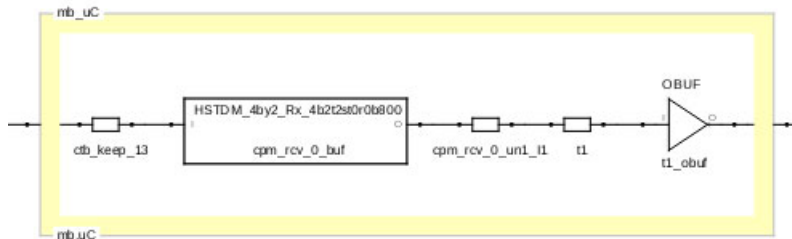
- mb\_uA (Sender)  
Clock-to-HSTDM constraints for each clock



```
# Time budgeting constraints for paths from user logic to HSTDM logic (transmit side)
# Constraints for Ratio 4
# TxUserLogic (2.7) + TxCombDelay (2.8) + SlackDis (18.0) = 23.4
set cpm_snd_0_data_0_c_clk1_r [expr 23.4 + $HSTDM4_v_snd_constraint_adjust]
set datapathonly_delay -comment {HSTDM tx budget HSTDM4_v (mb.uA:bank 27)->
  (mb.uC:bank 32)} -rise_from [get_clocks -include_generated_clocks {c:clk1}]
  -to {i:cpm_snd_0.bit_tx.mserdes} $cpm_snd_0_data_0_c_clk1_r
```

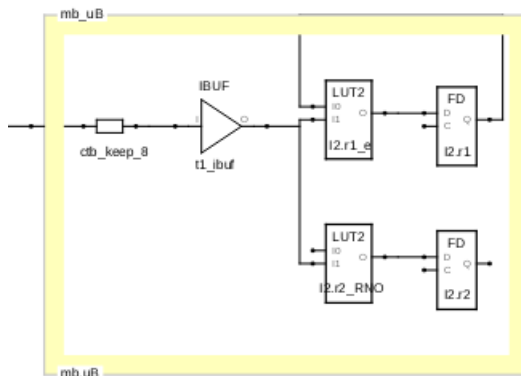
```
# TXUserLogic (3.5) + TxCombDelay (2.8) + SlackDis (18.3) = 24.6
set cpm_snd_0_data_0_c_clk2_r [expr 24.6 + $HSTDM4_v_snd_constraint_adjust]
set datapathonly_delay -comment {HSTDM tx budget HSTDM4_v (mb.uA:bank_27)->
    (mb.uC:bank_32)} -rise_from [get_clocks -include_generated_clocks {c:clk2}]
    -to {i:cpm_snd_0.bit_tx.mserdes} $cpm_snd_0_data_0_c_clk2_r
```

- **mb\_uC (Intermediate)**  
TDM-to-port DPO constraints; no output delays



```
# Time budgeting constraints for paths from user logic to HSTDM logic (receive side)
# Constraints for Ratio 4
# RxUserLogic (4.5) + RxCombLogic (0.8) + SlackDis (30.3) = 35.5
set cpm_rcv_0_data_0_p_t1 [expr 35.5 + $HSTDM4_v_rcv_constraint_adjust]
set datapathonly_delay -comment {HSTDM rx budget HSTDM4_v (mb.uA:bank_27)->
    (mb.uC:bank_32)} -from {i:cpm_rcv_0.rxbit.ISERDES_RX_DATA_00}
    -to {p:t1} $cpm_rcv_0_data_0_p_t1
```

- **mb\_uB (Receiver)**  
Port-to-clock constraints for each associated clock



```
# Time budgeting constraints for direct paths
# BudgetableDelay (1.2) + SlackDis (8.1) = 9.3
set datapathonly_delay -rise_to [get_clocks -include_generated_clocks {clk1}]
  -from {p:tl} 9.3
# BudgetableDelay (1.2) + SlackDis (8.1) = 9.3
set datapathonly_delay -rise_to [get_clocks -include_generated_clocks {clk2}]
  -from {p:tl} 9.3
```

Time Budgeting Calculation

See the following for details about how timing budgets are calculated:

- [Requested Period and Timing Budgets](#), on page 152
- [Timing Report Information for Timing Budgets](#), on page 153
- [Tx and Rx Budget Calculations](#), on page 154

Requested Period and Timing Budgets

The timing budgeter divides the total slack between the partitions, based on the estimated logic delay on the Tx (sender) and Rx (receiver) sides. On a path between partitions, the tool apportions delay as shown below. The colored boxes correspond to data in the excerpted timing report shown below, after the table.

Requested Period

= Tx\_budget + Rx\_budget + Board\_delay + Clk2Q + Setup - Clk\_skew

= 18.041 + 29.416 + 1.023 + 0.337 - 0.045 - (3.974 - 5.202)

= 50

Requested period	Reported in the path information section of the timing report.
Tx_budget + Rx_budget	<i>Total Slack</i> + <i>Total Estimated Logic Delay</i> The total slack is reported in the path information section of the timing report, and details of the delay estimates for the budgets are described in <a href="#">Tx and Rx Budget Calculations</a> , on page 154.
Board delay	Reported as delay in the detailed path information.



Clk2Q	Reported as delay in the detailed path information.
Setup	Reported as setup time in the path information section of the timing report.
Clk_skew	<i>(clock delay at end point) - (clock delay at start point)</i> These delays are reported in the path information section of the timing report.

## Timing Report Information for Timing Budgets

This is an excerpt from an example timing report, showing relevant values in the path information section. The colored boxes correspond to elements that make up the calculations described in [Requested Period and Timing Budgets](#), on page 152 and [Tx and Rx Budget Calculations](#), on page 154.

Path information for path number 1:

Requested Period:	50.00
- Setup time:	-0.045
+ Clock delay at ending point	3.974
= Required time:	54.019
- Propagation time:	9.554
- Clock delay at starting point:	5.202
= Slack (non-critical) :	39.263

Number of logic level(s)13

Starting point	mb.uA.arb_inst.curr_state[0] / Q
Ending point:	mb.uB.word_xfer_inst.data[1] / D
The start point is clocked by	mb_uB CLKOUT0_derived_clock_CLKIN1 [rising] on pin C
The end point is clocked by	gclk1_p [rising] on pin C

Instance / Net Name	Type	Pin Name	Pin Dir	Delay	Arrival Time	No. of Fan Out(s)
mb.uA.arb_inst.curr_state[0]	FDC	Q	Out	0.337	5.539	-
arb_inst.curr_state[0]	Net	-	-	0.442	-	5
mb.uA.arb_inst.curr_state_s3_0	LUT2	I0	In	-	5.981	-
mb.uA.arb_inst.curr_state_s3_0	LUT2	O	Out	0.150	6.131	-
N_52_1	Net	-	-	0.306	-	1
mb.uA.grant1_obuf	OBUF	I	In	-	6.437	-
mb.uA.grant1_obuf	OBUF	O	Out	2.217	8.654	-
grant1	Net	-	-	1.023	-	1
mb.uB.grant1_ibuf	IBUF	I	In	-	9.677	-
...						
mb.uB.word_xfer_inst.data[1]	FDC	D	In	-	14.756	-

## Tx and Rx Budget Calculations

The Tx and Rx budgets are apportioned based on the ratio of estimated logic delay on the Tx and Rx sides. The budgets are calculated as shown below, and are used as the values for the `set_datapathonly_delay` constraints.

$$\text{Tx | Rx budget} = \text{user\_logic\_delay} + \text{allocated\_slack}$$

$$\text{Tx budget} = 3.115 + 14.926 = 18.041$$

$$\text{Rx budget} = 5.079 + 24.337 = 29.416$$

The `user_logic_delay` is calculated individually for the Tx and Rx FPGAs, using the equation below. The arrival times are reported in the timing report; see the corresponding color-coded values reported in the excerpt.

$$\text{tx|rx\_user\_logic\_delay} = \text{arrival\_time\_end} - \text{arrival\_time\_start}$$

$$\text{Tx\_user\_logic\_delay} = 8.654 - 5.539 = 3.115$$

$$\text{Rx\_user\_logic\_delay} = 14.756 - 9.677 = 5.079$$

The following equation used to calculate allocated slack for Tx and Rx. The total slack is reported in the path information section of the timing report, and the board delay is reported in the path details (see the corresponding color-coded boxes in the timing report excerpt, above).

$$\text{allocated\_slack} = (\text{tx|rx\_delay} / (\text{total\_delay} - \text{boardDelay})) * \text{total\_slack}$$

$$\text{Tx\_allocated\_slack} = (3.115 / ((14.756 - 5.539) - 1.023)) * 39.263 = 14.926$$

$$\text{Rx\_allocated\_slack} = (5.079 / ((14.756 - 5.539) - 1.023)) * 39.263 = 24.337$$

## CHAPTER 5

# Inputs, Reports and Log Files

---

Reports and log files are available throughout the design cycle and on demand. Execution of a run or report command generates one or more reports. Reports are individually named and have various extensions such as .rpt, .srr, or .log.

- [Report and Log File Access](#), on page 156
- [Report View](#), on page 157
- [Database Reports and Log Files](#), on page 158
- [assign\\_clock](#), on page 143
- [UPF Report](#), on page 183
- [Formal Verification Report and Files](#), on page 186
- [Unified Compile Input Files](#), on page 188
- [Proto\\_rt Input Files](#), on page 193

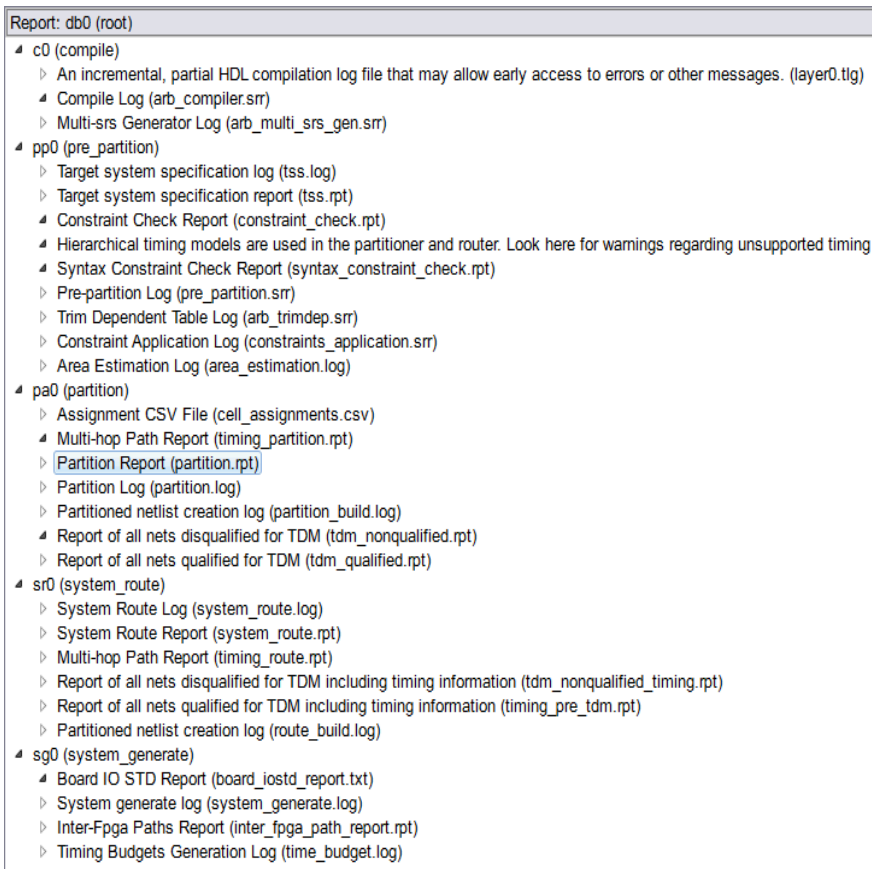
## Report and Log File Access

Report and log files can be accessed or viewed using any of the methods outlined in the table below (see [Checking Reports and Log Files](#), on page 512 for more information).

Report Viewing Method	Description and Reference Information
Log file viewer in GUI	Standard read-only report access through the GUI for the current state within the database. Includes hypertext links. This method allows browsing of all logs and reports for the database and supports the most functions such as cross-probing from the log file to the RTL or schematic.
HTML viewer	Read-only report access through any HTML browser outside of the database. Similar to the GUI method, provides a single page summary of all log files and reports with links to them. Allows viewing reports without consuming a license.
view report Tcl command	In GUI mode, opens the log file viewer GUI for read only. In shell mode, displays either all reports or a specified report in the current state. Use report list to view the available reports. Includes hypertext links.
export report Tcl command	Exports specified report in ASCII read-write mode to a location outside of the database. Use this method when you need access to the text version of a report, for example, when building a summary with a custom script. Use the auto_export_reports option to export text versions of log files and reports to an adjacent report database. Use report list to view the available reports.

# Report View

To view the reports and log files available within the current database state, use the report list command. Use the view report command and include the report name as an argument to view only the requested report. Omitting the report name argument lists all of the preceding reports up to and including the current database state and displays the first report listed. Use the browser section at the left to display the desired report on the right. The figure below shows the set of reports for a complete compile-to-system generate flow. Other database states such as pre-map, map, and instrumentation preparation include different report and log files.



The individual reports are described in [Database Reports and Log Files, on page 158](#).

## Database Reports and Log Files

The following sections summarize the available reports and log files, the report name and the corresponding name entered for the view report command argument, the report or run command responsible for generating the report, and a brief overview of the report content. The report commands use the current database state as input. Some reports are only valid from specific states. If a report is not valid from the current state, a list of valid states is be listed. The reports and log files are sectionalized as outlined below:

- [List of Report and Log Files](#), on page 159
- [Synthesis Database Report and Log Files](#), on page 160
- [Pre-Partition Database State Log Files and Reports](#), on page 163
- [Partition Database State Log Files and Reports](#), on page 165
- [System-Route Database State Log Files and Reports](#), on page 168
- [System-Generate Database State Log Files and Reports](#), on page 169
- [Miscellaneous Log Files and Reports](#), on page 171

## List of Report and Log Files

A complete, alphabetized list of all available reports and log files is presented in the following table.

<b>view report Command Argument</b>	<b>Name in Report View</b>
aptn.log	<a href="#">Automatic Partitioner (aptn) Log</a> , on page 171
archive_summary.log	<a href="#">Archive Summary Log</a> , on page 172
area_estimation.log	<a href="#">Area Estimation Log</a> , on page 165
board_iodstd_report.txt	<a href="#">Board IO STD Report</a> , on page 170
cell_assignments.csv	<a href="#">Assignment CSV File</a> , on page 166
constraints_application.srr	<a href="#">Constraint Application Log</a> , on page 165
constraint_check.rpt	<a href="#">Constraint Check Report</a> , on page 161
designName_compile.srr	<a href="#">Compile Log</a> , on page 161, <a href="#">RTL Diagnostics Report</a> , on page 171
designName_identify_db_generator.srr	<a href="#">Pre-Instrument Log</a> , on page 171
designName_multi_srs_gen.srs	<a href="#">Multi-srs Generator Log</a> , on page 161
designName_trimdep.srr	<a href="#">Trim Dependent Table Log</a> , on page 165
haps_io_report.txt	<a href="#">HAPS IO Report</a> , on page 162
map.srr	<a href="#">Synthesis Log</a> , on page 162
pcf_tss_report.rpt	<a href="#">Partition &amp; Route Target System Report</a> , on page 171
par_explorer.log	<a href="#">Exploratory PAR report</a> , on page 163
partition_build.log	<a href="#">Partition netlist creation log</a> , on page 167
partition.log	<a href="#">Partition Log</a> , on page 166
partition.rpt	<a href="#">Partition Report</a> , on page 167
partitioner_timing_model.log	<a href="#">Unsupported Timing Models Log</a> , on page 164
pre_map.srr	<a href="#">Pre-map Log</a> , on page 162
pre_partition.srr	<a href="#">Pre-partition Log</a> , on page 164

<b>view report Command Argument</b>	<b>Name in Report View</b>
route_build.log	<a href="#">Partition netlist creation log</a> , on page 169
system_generate.log	<a href="#">System generate log</a> , on page 170
system_route.log	<a href="#">System Route Log</a> , on page 168
system_route.rpt	<a href="#">System Route Report</a> , on page 168
ta.rpt	<a href="#">Timing Path Report</a> , on page 163
tdm_nonqualified.rpt	<a href="#">Report of all nets disqualified for TDM</a> , on page 167
tdm_qualified.rpt	<a href="#">Report of all nets qualified for TDM</a> , on page 167
time_budget.log	<a href="#">Timing Budgets Generation Log</a> , on page 170
timest_cons.log	<a href="#">Timing Estimation Constraint Application Log</a> , on page 170
timing.rpt	<a href="#">Multi-hop Path Report</a> , on page 166
tss.log	<a href="#">Target System Specification Log</a> , on page 164
tss.rpt	<a href="#">Target System Specification Report</a> , on page 164

## Synthesis Database Report and Log Files

Synthesis report and log files are generated during the compile, pre-map, and map database states (when the `design_flow` option is set to synthesis). Individual synthesis reports and log files can be explicitly displayed by entering the name string argument for the view report command listed in the following table.

<b>view report Command Argument</b>	<b>Name in Report View</b>
constraint_check.rpt	<a href="#">Constraint Check Report</a> , on page 161
<i>designName_compile.srr</i>	<a href="#">Compile Log</a> , on page 161
<i>designName_multi_srs_gen.srs</i>	<a href="#">Multi-srs Generator Log</a> , on page 161
haps_io_report.txt	<a href="#">HAPS IO Report</a> , on page 162



<b>view report Command Argument</b>	<b>Name in Report View</b>
syntax_constraint_check.rpt	<a href="#">Syntax Constraint Check Report</a> , on page 162
pre_map.srr	<a href="#">Pre-map Log</a> , on page 162
map.srr	<a href="#">Synthesis Log</a> , on page 162
ta.rpt	<a href="#">Timing Path Report</a> , on page 163
par_explorer.log	<a href="#">Exploratory PAR report</a> , on page 163

## Constraint Check Report

This report is available from a pre-map state or on demand by execution of a report constraint\_check command. Constraint checker can be run in the background using the **-bg** option of the command. The report is automatically displayed on command execution and can be redisplayed from a view report command by including a constraint\_check.rpt argument. An option (no\_constraint\_check\_in\_premap) can be used to disable constraint check report generation during the pre-mapper flow. The report lists clock relationships and constraint applicability; a summary is included at the beginning of the report.

## Compile Log

The run compile command produces a single report file (*designName\_compile.srr*) which can be displayed by a view report command entered from the compile (or subsequent) database state. The displayed report in the Report View is named Compile Log, and is shared by the report constraint\_check and report rtl\_diagnostics commands. The report lists the results of design compilation.

## Multi-srs Generator Log

The Multi-srs Generator log is available from a pre-partitioned (or subsequent) state following a run pre\_partition command. The file can be displayed with a view report command by including a *designName\_multi\_srs\_gen.srr* name argument entered from the pre-partition database state. The report lists netlist linker status.

## HAPS IO Report

The HAPS IO Report is available from a pre-map (or subsequent) state following a run `pre_map` command. The report can be displayed with a `view report` command by including a `haps_io_report.txt` name argument entered from the pre-map database state. The report lists the HAPS MGB, HapsTrak®, and UMRBus® board connector assignments and maps them to their corresponding Xilinx pin/bank/SLR. The individual GCLK and UMRBus pin assignments are also reported.

## Syntax Constraint Check Report

The Syntax Constraint Check Report is available from the pre-map or map database state or from the pre-partition state and produces a single report file (`syntax_constraint_check.rpt`). The report displayed in the Report View is named Syntax Constraint Check Report. The report file can be displayed with a `view report` command by including a `syntax_constraint_check.rpt` name argument. The report lists any constraint syntax violations and includes a clock summary.

## Pre-map Log

The Pre-map Log is available from a pre-partitioned (or subsequent) state following a run `pre_partition` command. The log file can be displayed with a `view report` command by including a `pre_map.srr` name argument entered from the pre-partition database state. The log lists the pre-map phase progress and includes both an IGC latch removal summary and sections with clock information such as a Clock Summary, Clock Load Summary, and Clock Optimization Report.

## Synthesis Log

The Synthesis Log is available from a mapped (or subsequent) state following a run `map` command. The log file can be displayed with a `view report` command by including a `map.srr` name argument entered from the mapped database state. The log file lists technology mapping and design optimizations, clock conversions and optimizations, and detailed timing information including worst-case slack. Device resource usage is reported in the Mapping Summary.

## Timing Path Report

The Timing Path Report is available from a mapped state on execution of a report timing -generate command. The report is automatically displayed and can be redisplayed from a view report command by including a `timn.ta.rpt` argument. Timing information in the report includes a performance summary and detailed timing on worst-case slack. From the timing report, clicking a View Worst Path in Analyst link opens the HDL Analyst with a filtered view of the corresponding path.

## Exploratory PAR report

The Exploratory PAR report is available from a mapped state following a run map command. The report can be displayed with a view report command by including a `par_explorer.log` name argument entered from the mapped database state. The report lists the maximum number of parallel place-and-route jobs to use.

## Pre-Partition Database State Log Files and Reports

The run `pre_partition` command produces a set of report log files that can be displayed by a view report command entered from the pre-partition (or subsequent) database state. Individual reports can be explicitly displayed by entering the name string argument for the view report command as shown in the following table:

view report Command Argument	Report Name in Report View
tss.log	<a href="#">Target System Specification Log</a> , on page 164
tss.rpt	<a href="#">Target System Specification Report</a> , on page 164
constraint_check.rpt	<a href="#">Constraint Check Report</a> , on page 161
partitioner_timing_model.log	<a href="#">Unsupported Timing Models Log</a> , on page 164
syntax_constraint_check.rpt	<a href="#">Syntax Constraint Check Report</a> , on page 162
pre_partition.srr	<a href="#">Pre-partition Log</a> , on page 164

<b>view report Command Argument</b>	<b>Report Name in Report View</b>
<code>designName_trimdep.srr</code>	<a href="#">Trim Dependent Table Log</a> , on page 165
<code>constraints_application.srr</code>	<a href="#">Constraint Application Log</a> , on page 165
<code>area_estimation.log</code>	<a href="#">Area Estimation Log</a> , on page 165

## Target System Specification Log

The Target system specification log is available from a pre-partitioned (or subsequent) state following a run `pre_partition` command. The log file can be displayed with a view report command by including a `tss.log` name argument entered from the pre-partition database state. The Target system specification log describes the results of board-system compilation and generation.

## Target System Specification Report

The Target system specification report can be displayed with a view report command by including a `tss.rpt` name argument entered from the pre-partition (or subsequent) database state. The report describes the composition of the target system specification in terms of board and cable usage and clock, reset, and voltage region settings.

## Unsupported Timing Models Log

The Unsupported Timing Models Log is available from a pre-partitioned (or subsequent) state following a run `pre_partition` command. The log file can be displayed with a view report command by including a `partitioner_timing_model.log` name argument entered from the pre-partition database state. The log file lists unsupported and hierarchical timing models targeted for the partitioner and router.

## Pre-partition Log

The Pre-Partition Log is available from a pre-partitioned (or subsequent) state following a run `pre_partition` command. The log file can be displayed with a view report command by including a `pre_partition.srr` name argument entered from the pre-partition database state. The log file describes the mapper initialization sequence and includes sections with clock information such as a Clock Summary, Clock Load Summary, and Clock Optimization Report.

## Trim Dependent Table Log

This log is available from the pre-partition state and can be displayed with a view report command by including a *designName\_trimdep.srr* name argument entered from the pre-partition database state. The log lists the linker status from generating the multi-SRS database.

## Constraint Application Log

The Constraint Application Log file is available from a pre-partitioned (or subsequent) state following a run *pre\_partition* command. The file can be displayed with a view report command by including a *constraints\_application.srr* name argument entered from the pre-partition database state. The report lists mapper initialization status.

## Area Estimation Log

The Area Estimation Log file is available from a pre-partitioned state following a run *pre\_partition* command. The log file can be displayed by a view report command entered from the pre-partition (or subsequent) database state. The individual log file can be explicitly displayed by including an *area\_estimation.log* name argument with the view report command. The log includes a mapping summary and a link to the *area\_est\_map.srr* report.

## Partition Database State Log Files and Reports

The run partition command produces a set of report log files that can be displayed by a view report command entered from the partition (or subsequent) database state. Individual partitioning reports can be explicitly displayed by entering the name string argument for the view report command as shown in the following table:

view report Command Argument	Report Name in Report View
cell_assignments.csv	<a href="#">Assignment CSV File</a> , on page 166
timing_partition.rpt	<a href="#">Multi-hop Path Report</a> , on page 166
partition.log	<a href="#">Partition Log</a> , on page 166
partition.rpt	<a href="#">Partition Report</a> , on page 167
partition_build.log	<a href="#">Partition netlist creation log</a> , on page 167

<b>view report Command Argument</b>	<b>Report Name in Report View</b>
tdm_nonqualified.rpt	<a href="#">Report of all nets disqualified for TDM</a> , on page 167
tdm_qualified.rpt	<a href="#">Report of all nets qualified for TDM</a> , on page 167
<topModuleName>_identify_report.log	<a href="#">Identify Instrumentor Report Log</a> , on page 167

## Assignment CSV File

The assignment CSV file reports the physical resources required by the design. The report is in Comma Separated Value format for spread-sheet display. To create a basic excel spread sheet:

1. Copy the report file contents, beginning with line 6, into an excel spread sheet.
2. On the Data tab, select Text to Columns to open the first sheet of the wizard and click Next.
3. On the second, make sure that the Comma delimiter check box is checked and click Next.
4. On the third sheet, view the preview and click Finish to update the spread sheet with the column-formatted data.

## Multi-hop Path Report

The Multi-hop Path Report can be displayed with a view report command by including a timing.rpt name argument entered from the partition (or subsequent) database state. The report describes the inter-FPGA critical net connections and multi-hop pad lengths.

## Partition Log

The Partition Log (partition.log) describes the partitioning flow from environment preparation through various summaries and finally the writing of results to the PCF file. The log file includes section and subsection headings identified by both title and an AP message ID; the message IDs are linked directly to the messages references documentation.

## Partition Report

The Partition Report (partition.rpt) is a detailed report of the configuration, target system detail including bin, port, trace, and cell assignments, and net and global routing information. The report file includes section and subsection headings identified by both title and an AP message ID; the message IDs are linked directly to the messages references documentation.

## Partition netlist creation log

The Partition netlist creation log is displayed by a view report command by including a partition\_build.log name argument entered from the partition (or subsequent) database state. The log reports the status of both the unpartitioned and post-partitioned netlists.

## Report of all nets disqualified for TDM

The Report of all nets disqualified for TDM is displayed by a view report command by including a tdm\_nonqualified.rpt name argument entered from the partition (or subsequent) database state. The report lists the nets that are not qualified for TDM.

## Report of all nets qualified for TDM

The Report of all nets qualified for TDM is displayed by a view report command by including a tdm\_qualified.rpt name argument entered from the partition (or subsequent) database state. The report lists the nets that are qualified for TDM.

## Identify Instrumentor Report Log

This report is generated only if you run the **report instrumentation** command.

The report provides the instrumentation details on each FPGA, based on the Instrumentor Debugger Constraints (IDC) files applied up to the point in the flow. This report helps you reduce the turnaround time by detecting debugging limitations early in the cycle.

## System-Route Database State Log Files and Reports

The run `system_route` command produces a set of report log files that can be displayed by a view report command entered from the system-route (or subsequent) database state. Individual reports can be explicitly displayed by entering the name string argument for the view report command as shown in the following table:

view report Command Argument	Report Name in Report View
timing_route.rpt	<a href="#">Multi-hop Path Report</a> , on page 168
system_route.log	<a href="#">System Route Log</a> , on page 168
system_route.rpt	<a href="#">System Route Report</a> , on page 168
route_build.log	<a href="#">Partition netlist creation log</a> , on page 169
<topModuleName>_identify_report.log	<a href="#">Identify Instrumentor Report Log</a> , on page 169

### Multi-hop Path Report

The Multi-hop Path Report can be displayed with a view report command by including a `timing.rpt` name argument entered from a partition, system-route, or subsequent database state. The report describes the inter-FPGA critical net connections and multi-hop pad lengths.

### System Route Log

The routing log file (`system_route.log`) describes the routing flow from preparation through various routing summaries including net paths, trace usage, and global routes, and concluding with the writing of results. The log file includes section and subsection headings identified by both title and an AP message ID; the message IDs are linked directly to the messages references documentation.

### System Route Report

The System Route Report (`system_route.rpt`) is a detailed report of the configuration and target system detail including individual:

- pin and trace assignments
- net and global routing information



- feedthrough and net-splitting information

The report file includes section and subsection headings. These sections are identified by both title and an AP message ID; the message IDs are linked directly to the messages references documentation.

## Partition netlist creation log

The Partition netlist creation log is displayed by a view report command by including a route\_build.log name argument entered from the partition (or subsequent) database state. The log reports the status of both the unpartitioned and post-partitioned netlists.

## Identify Instrumentor Report Log

This report is generated only if you run the **report instrumentation** command.

The report provides the instrumentation details on each FPGA, based on the Instrumentor Debugger Constraints (IDC) files applied up to the point in the flow. This report helps you reduce the turnaround time by detecting debugging limitations early in the cycle.

## System-Generate Database State Log Files and Reports

The run system\_generate command produces a set of report log files that can be displayed by a view report command entered from the system-generate database state. Individual reports can be explicitly displayed by entering the name string argument for the view report command as shown in the following table:

view report Command Argument	Report Name in Report View
board_iodtd_report.txt	<a href="#">Board IO STD Report</a> , on page 170
system_generate.log	<a href="#">System generate log</a> , on page 170
syntax_constraint_check.rpt	<a href="#">Syntax Constraint Check Report</a> , on page 170
timest_cons.log	<a href="#">Timing Estimation Constraint Application Log</a> , on page 170
time_budget.log	<a href="#">Timing Budgets Generation Log</a> , on page 170

## Board IO STD Report

The Board IO STD Report can be displayed with a view report command by including a `board_iostd_report.txt` name argument entered from the system-generate database state. The file lists the I/O used by board and their voltage levels.

## System generate log

The System Generate Log can be displayed with a view report command by including a `system_generate.log` name argument entered from the system-generate database state. The log describes the FPGA and UMRBus assignments, TDM assignments and lists the SLP machine-generated exit code.

## Pre-map Log

The Pre-map Log can be displayed with a view report command by including a `pre_map.srr` name argument entered from the system-generate database state. The log includes sections with clock information such as a Clock Summary, Clock Load Summary, and Clock Optimization Report.

## Syntax Constraint Check Report

The Syntax Constraint Check Report can be displayed with a view report command by including a `syntax_constraint_check.rpt` name argument entered from the system-generate database state. The report lists the results from the constraint checker and also includes a clock summary.

## Timing Estimation Constraint Application Log

This log is available from the system-generate state and can be displayed with a view report command by including a `timest_cons.log` name argument entered from the system-generate database state. The log lists the results from the syntax and constraint checkers and also includes a clock summary.

## Timing Budgets Generation Log

The Time Budgets Generation Log can be displayed with a view report command by including a `time_budget.log` name argument entered from the system-generate database state (the report is also available from a mapped state on execution

of a report timing -generate command). The report is automatically displayed and can be redisplayed from a view report command by including a *designName.ta* argument. Timing information in the report includes a performance summary and timing details on worst-case slack.

## Miscellaneous Log Files and Reports

### RTL Diagnostics Report

This report is available only from root on execution of a report rtl\_diagnostics command. The report is automatically displayed on command execution and can be redisplayed from a view report command by including a *designName\_compiler.srr* argument. The report lists problems that can impact design compilation.

### Automatic Partitioner (aptn) Log

The aptn log file can be displayed with a view report command by including an aptn.log name argument following execution of a report target\_system command. The log file lists the target system details for bins and connectivity. The report file includes section and subsection headings. These sections are identified by both title and an AP message ID; the message IDs are linked directly to the messages references documentation.

### Partition & Route Target System Report

The Partition & Route Target System Report can be displayed with a view report command by including a pcf\_tss\_report.rpt name argument following execution of a report target\_system command. The log file includes a target system summary and a target system details section. The report file includes section and subsection headings. These sections are identified by both title and an AP message ID; the message IDs are linked directly to the messages references documentation.

### Pre-Instrument Log

The Pre-Instrument Log can be displayed with a view report command by including a *designName\_identify\_db\_generator.srr* name argument entered from the pre-instrument database state. The log lists the results of design compilation.

## **Archive Summary Log**

The `archive_summary.log` file is written to the `.sar` destination file directory as a result of the archive operation. The summary lists the number of files archived and the archived database states.

## Configuration Commands for Partitioning Reports

This section describes the report-configuration commands that assist in the interpretation of the partition and system-route reports.

Many of the entries in the report files use PCF commands directly (see [Chapter 3, Partition Constraint File Tcl Commands](#) in the Command Reference). Other entries use “pseudo” commands, which cannot be direct inputs to the tool but which can be entered in specific files.

The reports rely on the command syntax to convey detailed configuration information. Reports can be extremely lengthy, but can be processed by scripts to include only the details of interest or to reformat for project requirements. For example, to list split nets that originate from a particular FPGA, you can create a script to list only the `modify_nets` commands that include `-split` and `-fromFPGA` arguments.

The following report-configuration commands are described in this section.

<a href="#">assign_trace</a>	<a href="#">global_route</a>	<a href="#">net_connections</a>
<a href="#">connection_model</a>	<a href="#">modify_net</a>	<a href="#">target_system</a>

A `report_control` PCF command is also available (see [report\\_control](#), on [page 192](#)) to control the type and amount of information printed to log and report files.

## connection\_model

Reports all of the available connection models for the current run.

### Syntax

**connection\_model** *modelType* **-modules** {*moduleList*}

*modelType*

The type of the connection model. Reported types are ACPM, HSTDm, HSTDm\_ERD, and DIRECT.

**-modules** {*moduleList*}

A list of the available multiplexing ratios for the *modelType*. The DIRECT connection model has a ratio of 1 (no multiplexing).

### Example

```
connection_model HSTDm -modules {  
    HSTDm_8by2 8  
    HSTDm_16by2 16  
    HSTDm_24by2 24  
    HSTDm_32by2 32  
    HSTDm_40by2 40  
    HSTDm_48by2 48  
    HSTDm_56by2 56  
    HSTDm_64by2 64  
    HSTDm_72by2 72  
    HSTDm_80by2 80  
    HSTDm_88by2 88  
    HSTDm_96by2 96  
    HSTDm_104by2 104  
    HSTDm_112by2 112  
    HSTDm_120by2 120  
    HSTDm_128by2 128  
}
```

## target\_system

Provides details of the target system for referencing information on pin and trace names. The target system report represents the most detailed target system after all PCF files have been applied.

### Syntax

```
target_system targetName
    bin binName -type typeName -pins pinCount [-locked]
        [-resource {resourceVector [ ... resourceVector] } ]
    pin pinName -bin owner -trace traceName -width width
        [-diff_pin pinName {-neg | -pos}] [-is_clock]
    trace traceName [-connection_model modelType]
        [-function {functionName}][-from name] -pins {pinNameList}
```

*targetName*

Specifies the name of the target system.

**bin** *binName* **-type** *typeName* **-pins** *pinCount* [**-locked**]  
 [**-resource** {*resourceVector* [ ... *resourceVector*] } ]

Specifies the name of the bin. The **-type** and **-pins** arguments are required and define the bin type (for example, Port or FPGA) and pin count. The optional **-locked** argument indicates that the bin has been locked either by default in the TSS generator or by the user. Cells and ports can only be assigned to locked bins through PCF commands. The **-resource** argument identifies any number of resource-value pairs enclosed in curly braces.

**pin** *pinName* **-bin** *owner* **-trace** *traceName* **-width** *width* [**-diff\_pin** *pinName* {**-neg** | **-pos**}]  
 [**-is\_clock**]

Specifies the name of the pin. The **-bin**, **-trace**, and **-width** arguments are required and identify the corresponding bin, the name of the associated trace, and the width of that trace. The optional **-diff\_pin** argument indicates if the pin is part of a differential pair (**-pos** is the default), and the **-is\_clock** argument identifies the pin as a clock pin.

**trace** *traceName* [**-connection\_model** *modelType*] [**-function** {*functionName*}]  
 [**-from** *name*] **-pins** {*pinNameList*}

Specifies the name of the trace. The **-pins** argument is required; the list entries are space-separated and enclosed in curly braces. The **-connection\_model**, **-function**, and **-from** arguments are optional and describe the model type (usually **DIRECT** to indicate that TDM is not allowed), trace function, and the source bin if it is a bidirectional trace.

## Example

```
target_system DETAIL

bin mb1.uA -type FPGA -pins 1123 -resource {LUT 1243200
    LUTM 621600 DFF 2486400 BRAM 1320 DSP 2160 IO 1200}
bin cpx1 -type Port -pins 50 -resource {PORT 50}
bin mb1.cde -type External -pins 105 -locked

pin mb1.cde/BC172.A_UMR_OUT[11] -bin mb1.cde -trace
mb1.A_UMR_OUT[11] -width 1 -diff_pin BC172.A_UMR_OUT[10] -neg

trace mb1.CLK_100MHz[0] -connection_model DIRECT
    -function {GCLK0_100MHz hard} -pins {
        mb1.clk.osc_100mhz/BC314.On
        mb1.uA/BC315.GCLKN[0]
        mb1.uB/BC316.GCLKN[0]
        mb1.uC/BC317.GCLKN[0]
        mb1.uD/BC318.GCLKN[0]
    }
```



## global\_route

Reports the global route result for each external net.

### Syntax

```
global_route netName [-feedthrough] [-unrouted -reason {descriptor}]  
  -tdm connectionModelName -ratio integer  
  -from binName [-through binName [... -through binName]] -to {binList}  
  [-using {traceGroupList}]
```

*netName*

The name of the external net.

#### **-feedthrough**

Indicates that the net is to be routed through one or more intermediate FPGAs.

#### **-unrouted**

Indicates that no legal routing could be found for the net (the **-using** field contains a list of trace groups with matching bin connectivity). The **-reason** argument indicates why the trace group is not legal. The reason *descriptors* are:

- **FUNCTION** (*ChannelFunction*) – the net and the channel do not have compatible function attributes. The reason includes the function of the channel in parentheses.
- **CAPACITY** – previous assignments to the channel have used all its capacity
- **CLOCK** – all clock resources for this channel have been used

**-tdm** *connectionModelName* **-ratio** *integer*

Indicates the TDM type and ratio.

**-from** *binName*

Lists the source bin for the net.

**-through** *binName*

Lists any intermediate bins between the **-from** bin and bins in the **-to** list.

**-to** *binList*

Lists one or more destination bins.

**-using** {*traceGroupList*}

Lists the trace groups used for the route.

## Report Examples

```
global_route clrn
  -from mb1.reset -to {mb1.uD mb1.uC mb1.uB mb1.uA mb1.cde mb2.uD
    mb2.uC mb2.uB mb2.uA mb2.reset mb2.cde} -using _T_5049

global_route net1
  -from mb_2.uA -to mb_1.uD -using _T_2198 -tdm HSTDm -ratio 24

global_route n[147]
  -feedthrough -from TOP_IO_HT3_mb1_D1 -to mb1.uA
  -through {mb1.uD} -using {_T_5047 _T_5034}

global_route -unrouted c[48]
```

## net\_connections

Reports limited information about the connections of the net. This section of the report also uses the `net_attribute PCF` command to report the non-default values of the net attributes and is paired with `global_route`.

### Syntax

```
net_connections netName -num_pins integer  
-source_port | -source_cell {binName cellName}  
-sink_port | -sink_cell {binName cellName} ...
```

*netName*

The name of the net.

**-num\_pins** *integer*

Lists the number of pins for the connection.

**-source\_port** | **-source\_cell** {*binName cellName*}

Reports the bin and cell of the driving port or cell for the net.

**-sink\_port** | **-sink\_cell** {*binName cellName*} ...

Reports a receiving bin and one cell or port in the receiving bin connected to the net. A cell or port is listed for each receiving bin connected to that net.

### Example

```
net_connections -num_pins 2 clk_3_p  
  -source_port {mb_1.clk.pll1 clk_3_p}  
  -sink_cell {mb_2.uA i_alpine_top}  
  
net_connections -num_pins 7 gclk0  
  -source_port {mb_1.clk.pll1 gclk0}  
  -sink_cell {mb_2.uA sync_dut_core_resetsn}  
  -sink_cell {mb_1.uC fpga_resetsn2}  
  -sink_cell {mb_1.uB fpga_resetsn3}  
  -sink_cell {mb_1.uA fpga_resetsn4}
```

## modify\_net

Reports the trace assignment for a net when multi-terminal and feedthrough nets are encountered. Each modified-net entry includes the original net name, the port bin of the driver, and the consuming port bin.

### Syntax

#### **modify\_net**

**-split -orig\_net** *netName* **-new\_net** *netName* **-fromFPGA** *binName*

**-feedthrough** *binName* **-orig\_net** *netName* **-in\_net** *netName* **-out\_net** *netName*

**-disconnect** *netName* **-connect** *netName* [**-toPort**|-**toFPGA**] *binName*

**-split -orig\_net** *netName* **-new\_net** *netName* **-fromFPGA** *binName*

Indicates a split (multi-terminal) net. The **-orig\_net** argument identifies the name of the original (pre-split) net, and the **-new\_net** argument identifies the name of the new net. The **-fromFPGA** argument defines the FPGA source of the original net.

**-feedthrough** *binName* **-orig\_net** *netName* **-in\_net** *netName* **-out\_net** *netName*

Indicates a net that passes through an intermediate FPGA (with no attached logic).

**-disconnect** *netName* **-connect** *netName* [**-toPort**|-**toFPGA**] *binName*

Indicates a net that is being relocated.

### Description

The partitioner creates buffered copies of nets when required to effectively use the available routing resources (traces). Three different net modifications are reported by **modify\_net**.

Net splitting copies a net so that it can be routed on a different channel from the original net and is reported using the **-split** option. For example, a net sourced in bin mb1.uA that drives cells in mb1.uB and mb1.uC must be split into two, point-to-point nets routed on the mb1.uA->mb1.uB and mb1.uA->mb1.uC traces.

Feedthroughs use an intermediate FPGA to enable a signal to be routed from its source FPGA to a destination FPGA when there are no available traces for a point-to-point route. Feedthrough paths are reported using the **-feedthrough**

option. For example, a net sourced in mb1.uA must drive a signal in mb1.uC, but there are no traces available between the two bins. The router, to make the connection, buffers the signal, assigns the buffer to an intermediate FPGA (mb1.uB), and creates a new net on the output of mb1.uB.

A new net resulting from a split or feedthrough operation must be connected to its proper destination bin and the original net must be disconnected. The -disconnect and -connect command options report the swap.

## System Route Report Example

```
modify_net -split -orig_net wbm_data_i8[24]
          -new_net wbm_data_i8_s[24] -fromFPGA mb1.uB

modify_net -feedthrough mb1.uB -orig_net wbm_addr_o8[31]
          -in_net wbm_addr_o8[31] -out_net wbm_addr_o8_ft[31]

modify_net -disconnect wbm_addr_o8[31]
          -connect wbm_addr_o8_ft[31] -toFPGA mb1.uC
```

## assign\_trace

Reports the detailed routing trace assignments.

### Syntax

```
assign_trace [-ratio ratio] [-module tdmModuleName] [-clock_capable]
netName{netList} traceName
```

#### **-ratio** *ratio*

Indicates the TDM ratio when the trace is multiplexed by an HSTDM or ACPM module. The *netList* argument lists all of the resultant multiplexed traces.

#### **-module** *tdmModuleName*

Indicates the TDM module type if the trace is being multiplexed.

#### **-clock\_capable**

Indicates if the trace is clock capable.

#### *netName*{*netList*}

Indicates the name of the net or nets assigned to the trace. The *netList* argument is a space separated list of all of the multiplexed nets assigned to the trace and is enclosed in curly braces.

#### *traceName*

The name of the assigned trace.

### System Route Report Example

```
assign_trace -ratio 4 -module HSTDM_4by2
{ newtop.t2.spc.tlu_trap_pc_0[46]
  newtop.t2.spc.tlu_trap_pc_1[15]
  newtop.t2.spc.tlu_npc_w[6]
  newtop.t2.spc.tlu_ifu_pstate_priv[0] } mb1_A23_D[5]

assign_trace wbm_data_o8[39] mb1_A23_C[11]
```

# UPF Report

The UPF report contains information about the number and names of power domains created along with their isolation and retention policies. It provides information about the power domain supply and corruption. See [Including UPF Specifications, on page 827](#) of the *User Guide* for information about using UPF specifications.

The UPF report contains information about the number and names of power domains created along with their isolation and retention policies. This report provides information regarding the power domain supply and corruption.

```

Report: db2 (root)
├─ c0 (compile)
│   └─ Compile Log (top_compiler.ssr)
│   └─ Multi-srs Generator Log (top_multi_srs_g
├─ pm1 (pre_map)
│   └─ HAPS IO Report (haps_io_report.txt)
│   └─ Syntax Constraint Check Report (syntax_
├─ Design Power Intent Information (upf.rpt)
│   └─ Constraint Check Report (constraint_che
│   └─ Pre-map Log (pre_map.ssr)
├─ m1 (map)
│   └─ Synthesis Log (map.ssr)
│       └─ Clock Optimization Summary
│           └─ 0 START OF TIMING REPORT
│               Mapping Summary:

```

```

=====
UPF Report
=====
Run Time : 0.004
Number of power domains : 2

1)
Name : PD1
Scope : top
Mapped : TRUE

Mapped elements : inst\[1\]\.inst_ram_wrap.inst_ram
Un-mapped elements :
Overlapping PDs :
Corruption : Disabled
Power supplies : primary-[ PD1.primary [power,PD1.primary.power] ]

Retention policies : 1
-----
1)Name : ret_PD1
Mapped : TRUE
Type : Domain
Save signal : {save,low}
Restore signal : {save,high}
Mapped elements : inst\[1\]\.inst_ram_wrap.inst_ram.dout[15:0] in
Un-mapped elements :

Isolation policies : 6
-----
1)Name : iso_PD1_both
Mapped : TRUE
Type : Domain
Control signal : {iso,high}
Clamp value : latch
Mapped elements : inst\[1\]\.inst_ram_wrap.inst_ram.dout[0] inst\[1\]
Un-mapped elements : inst\[0\]\.inst_ram_wrap.inst_ram.clk inst\[1\]\

```

```

=====
2)
Name : PD_top (Top power domain)
Scope : top
Mapped : FALSE

Mapped elements :
Un-mapped elements :
Overlapping PDs :
Corruption : Disabled
Power supplies : primary-[ PD_top.primary [power,PD_top.primary.power] ]

Retention policies : 0
-----
Isolation policies : 0
=====

```

See the following table for a detailed description of the UPF report.



Options	Description
Name: <i>name</i>	Specifies the power domain, isolation, or retention name.
Scope: <i>instance</i>	Specifies the instance in which the power domain was created.
Mapped: TRUE   FALSE	Specifies the status of the power domain, isolation policy, or retention policy. The power domain status overrides the status of the strategy. The status can be TRUE (implemented) or FALSE (ignored).
Mapped elements: <i>elementName</i>	Specifies the power domain, isolation, or retention elements that are marked for debug sampling.
Un-mapped elements: <i>elementName</i>	Specifies the power domain, isolation, or retention elements that are ignored.
Overlapping PDs: <i>PDnames</i>	Specifies information for the parent power domain.
Corruption: Enabled   Disabled	Specifies whether or not power domain corruption is enabled.
Power supplies: Primary-> <i>supplySet, ground, supplyNet, power, supplyNet</i>	Specifies information for the primary power supply of the power domain.
Type: Default   Explicit	Specifies if the isolation or retention strategy is domain-level (defined without -elements) or explicit (with -elements)
Save signal: <i>name, sense</i>	Specifies the retention save signal with either high or low sense.
Restore signal: <i>name, sense</i>	Specifies the retention restore signal with either high or low sense.
Control signal: <i>name, sense</i>	Specifies the isolation control signal with either high or low sense.
Clamp value: 0   1   latch	Specifies the isolation clamp value.

# Formal Verification Report and Files

A formal verification report is available as a result of running the Formality equivalency checker. Descriptions of the individual areas of the report are keyed to the table that follows.

Log Navigation - m6.vsl\_hdr\_fifo

Main Report Formality

Formality Information

Path:

/u/formal/nightly/synopsys

Run Directory:

/slowfs/sbg\_builds2/mandar/formalitytests/protoware/cisco\_reg/mydb/map/m6/verif/post\_map

Status:

Finished in 57.50 seconds

Post Map Formality Verification Summary (Status : Verification FAILED)

86.59% Design is Verified

Partition	Parameter	Verification Black Box	Area (% of the design)	Status	Hint	Report	Session
gtn_ig_txslice	more	more	---	Succeeded	---	gtnigtxslice_map.log	---
> txslice_sdr2ddr	more	more	---	Succeeded	---	txslicesdr2ddr_map1.log	---
> txslice_rm_vntag	more	---	13.42%	Failed	Resynthesize	txslicermvntag_map1.log	open
> txslice_rdr	more	---	27.27%	Succeeded	---	txslicerdr_map.log	---
> txslice_ODDRx32_txslice_lo32_u0	more	---	---	Succeeded	---	txsliceODDRx32txslic...log	---
> txslice_ODDRx32_txslice_hi32_u0	more	---	---	Succeeded	---	txsliceODDRx32txslic...log	---

Detailed Verification Report

Instantiation of modules containing inferred memories

Formality verification of memories is not currently supported

VCS Simulation (Beta)

Note:

1. Set the appropriate VCS\_HOME environment variable before running VCS simulation  
 2. Runs RTL Vs Synthesis netlist simulation for Verilog modules only  
 3. May need to modify the generated testbench and stimulus in some cases  
 4. Please click the VCS run link only once, and monitor the appropriate VCS run log

Instance	Resynthesize	
gtn_ig_txslice	run	<div>VCS Info</div> <div>Run directory: (Formality Run Dir)/formality_info/txslicercmerge/vcs_simulation</div> <div>Testbench: txslice_crc_merge_tb.v</div> <div>Stimulus: txslice_crc_merge.slm</div> <div>Run log: vcs.log</div> <div>(Run Directory)/runvcs</div>

Verification Black Box Report

Instance	Note
gtn_ig_txslice	User defined synthesis black box. Can not be verified.
> hfifo	
gtn_ig_txslice	User defined synthesis black box. Can not be verified.
> pfifo	

In the report:

**1****Formality Information**

Provides information regarding formality settings, formality working directory paths, and the total runtime taken for the complete verification process.

**Post Map Formality Verification Summary****2**

Provides overall information about the design being verified including:

- Partitions which were run.
- Parameter of partitions.
- Verification black-box information.
- Area of the partition.
- Status of the verification run for the partition.
- Hint which allows the user to re-synthesize the partition.
- Report Formality log file for a given partition.
- Session links for opening Formality session for the partitions which failed.

**Instantiation of Modules Containing Inferred Memories****3**

Lists all the design partitions that contain inferred memories. These cannot be verified using Formality and require VCS simulation.

- Instance: module instances that must be verified by simulation.
- Resynthesize: provides a link for re-synthesizing the partition for further analysis and debugging.
- VCS Simulation: launch VCS simulator.

**Verification Black-Box Report****4**

Lists all modules defined as synthesis black boxes originating from files other than Verilog and VHDL.

The Formality equivalency checker requires a Setup Verification for Formality (.svf) script file that contains the Formality commands. These commands provide guidance so that the Formality tool can account for the transformations made by the synthesis optimizations. For example, if the synthesis tool duplicates a register to improve drive strength, this file records the register duplication. From the information in this file, the Formality tool is able to account for the extra register during compare-point matching and verification. The SVF file supports the following Formality commands:

```
guide_reg_merging
guide_reg_duplication
guide_reg_constant
```

# Unified Compile Input Files

This section describes some files and associated commands used for unified compile:

- [UPF File for Unified Compile](#), on page 188
- [UTF File](#), on page 188
- [VCS Script](#), on page 191

## UPF File for Unified Compile

The IEEE 1801-2009 Unified Power Format (UPF) is a standard that allows you to focus on power as a key consideration early in the design process. Expressing power intent is increasingly important as larger designs, especially SoC designs, consume more power. The power specifications are provided in a UPF file, which can be used by multiple tools.

With the UC flow, the UPF file is specified in the VCS command line when the design is first compiled. This is different from the standard compiler flow where the file is read in and used at the run compile state.

- You can only specify one UPF file. If you have multiple files, specify one file with the `vcs` command and source the other files from this file.
- To control the value of corruption (register + memory) at compile time, specify this command in the UPF file:

```
set_design_attributes -attribute SNPS_random_corruption 0 | 1 | I
```

0 is the default value and specifies all zeros corruption. 1 specifies all ones corruption, and I specifies inversion-based corruption.

- Currently, you can only specify a UPF file for the following technologies: Xilinx Virtex UltraScale+ and Versal

## UTF File

The UTF (Unified Tcl Format) file is an input file for VCS unified compile, and contains design information. The file must include the `vcs_exec_command` to specify the VCS script, but others are optional. The UC methodology does not

support every UTF command; the following table lists some of the supported UTF commands that are most used. For details about the UTF commands, refer to the VCS documentation.

<code>vcs_exec_command {VCS_script}</code>	Required. Within the curly braces, specify the VCS script (see <a href="#">assertion_synthesis Syntax</a> , on page 74) that contains the commands to run the VCS tool. The script can be any supported VCS format, like .csh, .ksh, etc. Syntax example: <code>vcs_exec_command {vcs_dut.csh}</code>
<code>set_hwtop -module &lt;top_module_name&gt; vcs_exec_command -hw_top</code>	Specifies the top level of the design; can also be set through <code>vcs_exec_command</code> :
<code>synthesis -blackbox {moduleName}</code>	Black-boxes the specified modules in the designs, but you must include the definitions for the modules or entities in the VCS script. Use spaces to separate multiple module names. <code>synthesis -blackbox {andGate1 andGate2 andGate3}</code>
<code>assertion_synthesis</code>	Writes HDL assertions into the VCS database instead of ignoring them. Use <code>assertion_synthesis -enable</code> to write out all assertions. To enable specific assertions, use <code>assertion_synthesis +assert moduleName.label</code> . Here, <i>moduleName</i> and <i>label</i> identify the module and the assertion to enable, respectively. See <a href="#">assertion_synthesis Syntax</a> , on page 74 for an overview.
<code>memory_preferences scan_path {listofPaths}</code>	Specifies the search path for the memory initialization files (specified using \$readmemb/\$readmemh) in the UC flow. To specify multiple paths separate them using spaces. For example: <code>memory_preferences -scan_path {/prj/tools/ip2/memblock1 /prj/tools/ipe/memblock3}</code>

## Example UTF File

```
vcs_exec_command { vcs -full64\  
-hw_top=example_cgmac_wrapper \  
+libext+.v \  
+define+$CGMAC_ITF \  
../src/dut/example_cgmac_wrapper.v \  
../src/dut/dut.v }
```

## assertion\_synthesis Syntax

The UTF `assertion_synthesis` command controls how SVAs are implemented. Refer to the VCS documentation for the complete syntax.

```
assertion_synthesis [-enable <value>] [-ignore (<sva_type>)] [-path <path>]
                    [-tree <hierarchy_name>] [-module <module_name>] [-assert <#SVA_instance>]
```

### assertion\_synthesis Option      Description

-enable ALL	Synthesizes all assertions.
-ignore IMMEDIATE   CONCURRENT   ALL	Specifies the type of assertions to ignore. ALL means that no SVAs are synthesized.
-path <path>	Does not synthesize SVAs in the given hierarchy.
+path <path>	Synthesizes all SVAs in the specified hierarchy.
-tree <hierarchy_name>	Does not synthesize SVAs below the given hierarchy.
+tree <hierarchy_name>	Synthesizes all SVAs below the given hierarchy.
-module <module_name>	Does not synthesize any SVAs in the given module.
+module <module_name>	Synthesizes all SVAs in the given module.
-assert #SVA_instance	Does not synthesize the specified SVA instance.
+assert #SVA_instance	Synthesizes the specified SVA instance.

## wire\_resolution Command for XMR Conflicts

The UTF `wire_resolution` command is used to resolve conflicts when there are multiple drivers. This is the RTL code:

```
assign top.inst1.din1=din3;
sub1 #(.width(width)) inst1
(.clk(clk),.rst_n(rst_n),.din1(din1),.dout1(dout1));
```

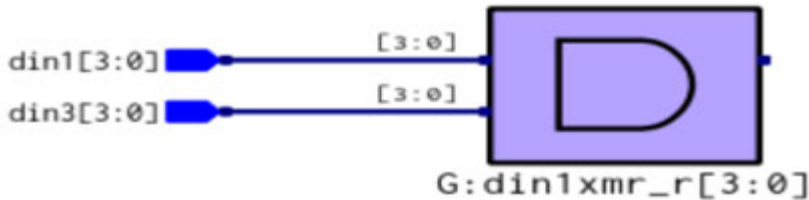
This is the UTF syntax:

```
wire_resolution -default_xmr_conflict {WAND | WOR | XMR}
```

The following examples show the effects of different `wire_resolution` command settings:

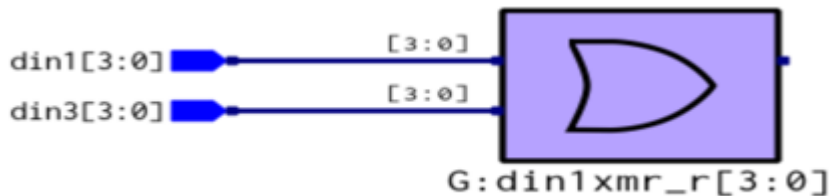
- Implementation when `wire_resolution` is set to the default, WAND:

```
wire_resolution -default_xmr_conflict {WAND}
```



- Implementation when `wire_resolution` is set to WOR:

```
wire_resolution -default_xmr_conflict {WOR}
```



- When `wire_resolution` is set to XMR, the XMR has the highest priority.

## VCS Script

The VCS script specifies the commands used to run unified compile on the design. This section contains an overview of typical commands; consult the VCS documentation for complete details.

The script must include this information:

- Basics
  - The RTL source files
  - The source language
  - The compilation library
  - The design top
  - Top-level VHDL generics and Verilog parameters

- include paths
- Top-level Verilog macros
- Library paths
- Verilog library file name extension
- The `vcs_exec_command`, which specifies VCS commands for analysis and elaboration.
  - `vlogan/vhdlan` analysis commands, according to the HDL used for the design, The table lists commonly-used options:

Argument	Description
<b>Verilog Analyzer (vlogan)</b>	
<code>+define+&lt;macro&gt;</code>	Defines a macro in the Verilog source
<code>-f file</code>	Specifies files and command options
<code>-l &lt;logfile&gt;</code>	Generates the log file
<code>-q</code> or <code>- quiet</code>	No internal messages and banner
<code>-v &lt;lib_file&gt;</code>	Specifies a Verilog library file
<code>-y &lt;libdir&gt;</code>	Specifies a directory of Verilog library files
<code>+libext+&lt;ext&gt;</code>	Specifies library file extensions (used with <code>-y</code> )
<code>-work &lt;libdir&gt;</code>	Analyzes into a specified logical library
<code>+v2k</code>	Enables Verilog 2001 constructs
<code>-sverilog</code>	Enables SystemVerilog constructs
<code>-timescale=1ns/1ps</code>	Specifies a default timescale
<code>+incdir+&lt;dir&gt;</code>	Specifies search directory for included files <code>+librescan</code> : Searches unresolved module starting first library in the <code>vlogan</code> command
<b>VHDL Analyzer (vhdlan)</b>	
<code>-work &lt;libdir&gt;</code>	Analyzes into a specified logical library
<code>-q</code> or <code>- quiet</code>	No internal messages and banner
<code>-vhdl87</code>	Enables VHDL 87 syntax instead of VHDL 93



Argument	Description
-f file	Specifies files and command options
-xlrn	Allows a relaxed or non-LRM-compliant code
-smart_order	Identifies the file order dependencies

- vcs elaboration command, to build the design hierarchy from the library files generated during the analysis phase. The table below lists commonly used arguments.

Argument	Description
Elaboration command (vcs)	
-l <logfile>	Generates the log file
-P pli.tab	Compiles the user-defined PLI table
<.c .o files>	Adds C or object files to compile
-xlrn	Allows a relaxed/non-LRM compliant code
-ignore_driver_checks	Suppresses multiple driver checks
-liblist:	Specifies the library search order for unresolved
module or entity	Instantiated in Verilog

Use these additional vcs commands to get information:

Command	Description
vcs -full64 -id	Checks the VCS setup, version, and platform
vcs -full64 -doc	Accesses VCS setup and compilation commands%

## Proto\_rt Input Files

This section describes some files and associated commands used by the Proto\_rt Confpro package to define HAPS modules for IP Training:

- [Proto\\_rt Tcl Script](#), on page 194

- [Hardware Mapping File \(HMF\)](#), on page 194

## Proto\_rt Tcl Script

You can run the `proto_rt` package from the host machine where the HAPS systems are connected. Call the `proto_rt` package from the Confpro shell.

You can run the script from the Confpro shell directly or from a TCL file. The first entry in the file must be `package require proto_rt`, followed by the `proto_rt` commands you want to run. You can then source the file in Confpro:

```
$HAPS_INSTALL_DIR/bin/confprosh run_commands.tcl
```

See [proto\\_rt::run\\_ipinfra, on page 96](#) in the *HAPS Prototyping Debugging Environment Reference* for syntax details for the commands to include in the script.

## Example

This is an example of a `run_commands.tcl` file:

```
package require proto_rt
proto_rt::run_ipinfra -hmf hmf.hmf -train all
```

## Hardware Mapping File (HMF)

To specify IP training with HAPS-100 systems, use an HMF file in JSON format with `.hmf` as the file name extension. The file specifies the serial numbers of the HAPS modules.

You can specify selected FPGAs from HAPS modules to be used for training. If no FPGA is specified, all FPGAs in the modules are considered for training. The HMF file is specified with the `proto_rt` command, which is used to run training. See [proto\\_rt::run\\_ipinfra, on page 96](#) in the *HAPS Prototyping Debugging Environment Reference* for the syntax of the command that uses the HMF information.

## Examples

1. Example of an HMF file with serial numbers of HAPS modules. All FPGAs in both the modules are considered for training.

```
{ "
  "tsdmaphaps": { "
    "FB1": { "serial": "X001234" },
    "FB2": { "serial": "X005678" }
  }
}
```

2. Example of an HMF file with serial number and FPGA information. Only FPGA A and B in both modules are used for the training.

```
{ "
  "tsdmaphaps": { "
    "FB1.uA": { "serial": "X001234", "fpga": "uA" },
    "FB1.uB": { "serial": "X001234", "fpga": "uB" },
    "FB2.uA": { "serial": "X005678", "fpga": "uA" },
    "FB2.uB": { "serial": "X005678", "fpga": "uB" }
  }
}
```

While using a HMF file, make sure all HAPS modules connected to the host are closed using the `cfg_close` command.



# Index

---

## Symbols

53

? wildcard  
Timing Analyzer 55

## A

area estimation log 165  
ASICs, converting  
    power considerations 144  
assertion\_synthesis 190  
assign\_trace command string 182

## B

bus bundling 76  
buses  
    compressed display 76  
    enabling bit range display 75  
    hiding in flattened Technology views 77

## C

cell interior display,  
    enabling/disabling 76  
clock alias 52  
commands  
    pseudo 173  
    report-configuration 173  
Compact View 73  
compiler  
    compared to unified compiler 142  
Configure IICE dialog box 134  
connection\_model command string 174  
connectivity, enabling bit range  
    display 75

## D

data view  
    operations 88  
database  
    creating 27  
    state manipulation 12  
database structure 11  
Dataflow View 73  
Debug, Verdi 137  
design templates 27  
dialog boxes  
    Configure IICE 134  
disqualified nets report 167  
docking windows 26

## F

filters  
    timing reports 50  
Floating License Usage command 84  
FPGA Implementation Tools  
    command 83  
from points  
    custom timing report 52  
    object search order (Timing  
        Analyzer) 53

## G

global\_route command string 177  
Go to SolvNet command 83  
graphical interface 26

## H

Help command 83  
Help menu 83  
hierarchy browser

enabling/disabling display 76  
HMF 194  
How to Use Help command 83

## I

IICE parameters  
individual 134  
instances  
expansion maximum limit 77  
expansion maximum limit (per filtered sheet) 80  
expansion maximum limit (per unfiltered sheet) 80  
name display 75  
isolation power strategy 144

## L

labels, displaying 75  
license  
floating 84  
License Agreement command 84  
Limit Number of Paths 52  
list of log files 159  
list of reports 159  
log file list 159  
log files 155  
logs  
area estimation 165  
map 162  
partition 166  
partition database 163, 165  
partition netlist creation 167, 169  
pre-map 162  
system route 168  
system-route database 168  
target system specification 164  
viewing 155

## M

map log 162  
menus  
Help 83  
modify\_net command string 180

multi-IICE  
tabs 134

## N

net\_connections command string 179

## O

object search order (Timing Analyzer) 53  
objects  
displaying compactly 76  
Online Documents command 83

## P

P&R Timing and Correlation 42  
partition database logs 163, 165  
partition database reports 163, 165  
partition log 166  
partition netlist creation log 167, 169  
partition report 167  
pins  
displaying names 75  
maximum on schematic sheet 80  
ports  
displaying names 75  
power domains, UPF 144  
prefixes  
timing analyst 52  
pre-map log 162  
primitives  
internal logic, displaying 76  
pseudo commands 173

## Q

qualified nets report 167

## R

report list 159  
Report View 157  
report-configuration commands 173  
reports

- disqualified nets [167](#)
- formality [186](#)
- partition [167](#)
- partition database [163](#), [165](#)
- qualified nets [167](#)
- system-generate database [169](#)
- system-route database [168](#)
- timing path [163](#)
- viewing [155](#)
- Resource Center
  - See Technical Resource Center
- retention power strategy [145](#)
- RTL diagnostics report
  - reports
    - RTL diagnostic [171](#)

## S

- schematic objects
  - displaying compactly [76](#)
- schematics
  - displaying labels [75](#)
  - sheet connectors [76](#)
- scripts [27](#)
  - design flow [17](#)
- Setup, Verdi Debug [137](#)
- sheet connectors [76](#)
- slack
  - slack margin [52](#)
- SolvNet Support command [82](#)
- svf script [187](#)
- symbols
  - enabling name display [75](#)
- Synopsys FPGA implementation tools
  - product information [83](#)
- Synopsys FPGA products [83](#)
- Synopsys Home Page command [83](#)
- Synopsys Training Page command [83](#)
- Synthesis Timing and Correlation [42](#)
- system route log [168](#)
- system-generate database reports [169](#)
- system-route database logs [168](#)
- system-route database reports [168](#)

## T

- target system specification log [164](#)
- target\_system command string [175](#)
- TCL Help command [84](#)
- Tcl scripts [27](#)
- Technical Resource Center
  - accessing [83](#)
- templates [27](#)
- through points
  - specifying for timing report [51](#)
  - timing analyst [51](#)
- timing analyst
  - description [48](#)
- timing analyzer
  - wildcards [55](#)
- timing path report [163](#)
- timing report
  - defining through points [51](#)
  - using path filtering [52](#)
- Timing Report View [40](#)
- timing reports
  - clock alias in custom report [52](#)
  - filters for custom reports [52](#)
  - from/to/through for custom reports [50](#)
- to points [53](#)
  - custom timing report [52](#)

## U

- unified compile
  - compared to native compiler [142](#)
- updates from the Resource Center [83](#)
- UPF
  - specifying isolation cells [144](#)
  - specifying power domains [144](#)
  - specifying retention registers [145](#)
- UTF
  - description [188](#)
  - example file [189](#)

## V

- vcs elaboration command [193](#)
- VCS script [191](#)

`vcs_exec_command` [192](#)  
Verdi Debug Setup [137](#)  
version information [84](#)  
`vhdlan` [192](#)  
`vlogan` [192](#)

## **W**

web updates [83](#)  
wildcards  
    timing analyzer [55](#)  
windows  
    docking [26](#)  
wire\_resolution, UTF [190](#)

## **X**

XMRs  
    UTF [190](#)