

Synopsys Synplify Pro for Microsemi Edition

Command Reference

November 2016



Copyright Notice and Proprietary Information

© 2016 Synopsys, Inc. All rights reserved. This software and documentation contain confidential and proprietary information that is the property of Synopsys, Inc. The software and documentation are furnished under a license agreement and may be used or copied only in accordance with the terms of the license agreement. No part of the software and documentation may be reproduced, transmitted, or translated, in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without prior written permission of Synopsys, Inc., or as expressly provided by the license agreement.

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

November 2016

Contents

Chapter 1: Introduction

About Tcl Commands	8
Tcl Conventions	8
Tcl Scripts and Batch Mode	8
About the GUI Commands	10
Graphic User Interface Commands	10
Tcl Commands	12
Document Set	12

Chapter 2: Tcl Commands

Alphabetical List of Commands	14
add_file	17
add_folder	21
analyst	21
check_fdc_query	24
command_history	26
constraint_file	26
create_fdc_template	27
design	28
encryptIP	36
encryptP1735	40
get_env	42
get_option	42
hdl_define	43
hdl_param	44
impl	45
job	46
log_filter	47
log_report	48
message_override	49
open_design	50

open_file	50
partdata	51
program_terminate	52
program_version	52
project	53
project_data	60
project_file	61
project_folder	62
recording	63
report_clocks	64
run_tcl	65
sd2fdc	65
set_option	66
status_report	91
syn_connect	94
syn_create_err_net	95
synplify_pro	97
Tcl Command Categories	100

Chapter 3: Tcl Find, Expand, and Collection Commands

find	102
Tcl Find Syntax	103
Tcl Find Syntax (Beta)	109
Tcl Find Syntax Examples	112
find -filter	116
expand	123
Collection Commands	126
c_diff	127
c_info	128
c_intersect	128
c_list	129
c_print	130
c_symdiff	130
c_union	131
define_collection	132
define_scope_collection	133
get_prop	134
set	134
Query Commands	136
all_clocks	139

all_fanin	139
all_fanout	141
all_inputs	143
all_outputs	144
all_registers	144
get_cells	146
get_clocks	148
get_clock_source	149
get_nets	150
get_pins	152
get_ports	154
object_list	155
report_timing	156
Synopsys Standard Collection Commands	160
add_to_collection	160
append_to_collection	162
copy_collection	163
foreach_in_collection	164
get_object_name	165
index_collection	166
remove_from_collection	167
sizeof_collection	169

Chapter 4: User Interface Commands

File Menu	172
New Command	173
Create Image Command	174
Build Project Command	176
Open Project Command	176
Edit Menu	177
Find Command (Text)	179
Find Command (In Project)	181
Find Command (HDL Analyst)	183
Find in Files Command	187
Replace Command	189
Goto Command	190
View Menu	191
Toolbar Command	194
View Sheets Command	195
View Log File Command	196

Project Menu	199
Add Source File Command	200
Remove Implementation	203
Change File Command	204
Set VHDL Library Command	204
Add Implementation Command	205
Convert Vendor Constraints Command	205
Archive Project Command	206
Un-Archive Project Command	207
Copy Project Command	210
Hierarchical Project Options Command	213
Implementation Options Command	214
Device Panel	215
Options Panel	217
Constraints Panel	218
Implementation Results Panel	220
Timing Report Panel	222
High Reliability Panel	223
Verilog Panel	224
VHDL Panel	228
Compiler Directives and Design Parameters	231
Push Tristates Option	240
Place and Route Panel	242
Import Menu	243
Run Menu	244
Run Tcl Script Command	247
Run Implementations Setup Command	248
Job Status Command	250
Identify Instrumentor Command	250
Launch Identify Debugger Command	252
Launch SYNCORE Command	252
SYNCORE FIFO Wizard	254
SYNCORE RAM Wizard	264
SYNCORE Byte-Enable RAM Wizard	268
SYNCORE ROM Wizard	271
SYNCORE Adder/Subtractor Wizard	275
SYNCORE Counter Wizard	279
Configure and Launch VCS Simulator Command	281
Analysis Menu	291
Timing Report Generation Parameters	292

HDL Analyst Menu	303
HDL Analyst Menu: RTL and Technology View Submenus	303
HDL Analyst Menu: Hierarchical and Current Level Submenus	304
HDL Analyst Menu: Filtering and Flattening Commands	306
HDL Analyst Menu: Timing Commands	310
HDL Analyst Menu: Analysis Commands	310
HDL Analyst Menu: Selection Commands	314
HDL Analyst Menu: FSM Commands	314
Options Menu	315
Configure Compile Point Process Command	316
Project View Options Command	319
Editor Options Command	324
Place and Route Environment Options Command	327
Project Status Page Location	327
New HDL Analyst Options Command	329
HDL Analyst Options Command	330
Configure External Programs Command	339
Web Menu	340
Help Menu	341
Preferred License Selection Command	342
Tip of the Day Command	343

Chapter 5: GUI Popup Menu Commands

Popup Menus	346
Watch Window Popup Menu	346
Tcl Window Popup Menu	347
Text Editor Popup Menu	347
Log File Popup Menu	347
FSM Viewer Popup Menu	350
Project View Popup Menus	353
Project Management View Popup Folder Commands	357
File Options Popup Menu Command	359
Copy File Popup Menu Command	361
Change Implementation Popup Menu Commands	361
Show Compile Points Popup Menu Command	362
Project Options Popup Menu Command	362
Add P&R Implementation Popup Menu Command	363
Options for Place & Route Jobs Popup Menu Command	366
RTL and Technology Views Popup Menus	368

CHAPTER 1

Introduction

This document is part of a set that includes reference and procedural information for the Synopsys[®] Synplify Pro[®] FPGA synthesis tool.

This document describes the commands available in the synthesis tools.

This chapter includes the following introductory information:

- [About Tcl Commands, on page 8](#)
- [About the GUI Commands, on page 10](#)
- [Document Set, on page 12](#)

About Tcl Commands

Tcl (Tool Command Language) is a popular scripting language for controlling software applications. Synopsys has extended the Tcl command set with additional commands that you can use to run the Synopsys FPGA programs. These commands are not intended for use in controlling interactive debugging, but you can use them to run synthesis multiple times with alternate options to try different technologies, timing goals, or constraints on a design.

Tcl scripts are text files that have a `tcl` file extension and contain a set of Tcl commands designed to complete a task or set of tasks. You can also run Tcl scripts through the Tcl window (see [Tcl Script Window, on page 48](#)).

The Synopsys FPGA Tcl commands are described here. For information on the standard Tcl commands, syntax, language, and conventions, refer to the Tcl online help (Help->TCL Help).

Tcl Conventions

Here is a list of conventions to respect when entering Tcl commands and/or creating Tcl scripts.

- Tcl is case sensitive.
- Comments begin with a hash mark or pound sign (#).
- Enclose all path names and filenames in double quotes ("").
- Use a forward slash (/) as the separator between directory and path names (even on the Microsoft® Windows® operating system). For example:

```
designs/big_design/test.v
```

Tcl Scripts and Batch Mode

For procedures for creating Tcl scripts and using batch mode, see [Working with Tcl Scripts and Commands, on page 522](#) in the *User Guide*:

- [Running Batch Mode on a Project File, on page 516](#)
- [Running Batch Mode with a Tcl Script, on page 517](#)
- [Generating a Job Script, on page 523](#)

- [Creating a Tcl Synthesis Script, on page 524](#)
- [Using Tcl Variables to Try Different Clock Frequencies, on page 526](#)
- [Running Bottom-up Synthesis with a Script, on page 528](#)

About the GUI Commands

The FPGA synthesis tools include a graphical user interface (GUI) as well as a command line capability. Most commands have GUI and command line versions, so you can use either method to specify commands.

The commands that are available vary with the capabilities of synthesis tools. The following sections give you an overview of the commands in tools:

- [Graphic User Interface Commands, on page 10](#)
- [Tcl Commands, on page 12](#)

Graphic User Interface Commands

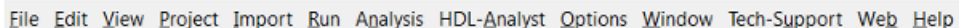
The GUI commands are accessed from the software graphical interface. Most commands open dialog boxes where you can specify parameters for the command.

The GUI provides a few ways to access commands:

- [Menus, on page 10](#)
- [Context-sensitive Popup Menus, on page 11](#)
- [Toolbars, on page 11](#)
- [Keyboard Shortcuts, on page 11](#)
- [Buttons and Options, on page 12](#)
- [Tcl Commands, on page 12](#)

Menus

The set of commands on the pull-down menus in the menu bar varies depending on the view, design status, task to perform, and selected object(s). For example, the File menu commands in the Project view differ slightly from those in the RTL view. Menu commands that are not available for the current context are dimmed out. The menu bar in the Project view is shown below:



The individual menus, their commands, and the associated dialog boxes are described in the following sections:

- [File Menu, on page 172](#)
- [Edit Menu, on page 177](#)
- [View Menu, on page 191](#)
- [Project Menu, on page 199](#)
- [Import Menu, on page 243](#)
- [Run Menu, on page 244](#)
- [Analysis Menu, on page 291](#)
- [HDL Analyst Menu, on page 303](#)
- [Options Menu, on page 315](#)
- [Web Menu, on page 340](#)
- [Help Menu, on page 341](#)

Context-sensitive Popup Menus

Popup menus, available by right-clicking, offer access to commonly used commands that are specific to the current context. See [Popup Menus, on page 346](#), [Project View Popup Menus, on page 353](#), and [RTL and Technology Views Popup Menus, on page 368](#) for information on individual popup menus.

Toolbars

Toolbars contain icons associated with commonly used commands. For more information about toolbars, see [Toolbars, on page 76](#).

Keyboard Shortcuts

Keyboard shortcuts are available for commonly used commands. The shortcut appears next to the command in the menu. See [Keyboard Shortcuts, on page 84](#) for details.

Buttons and Options

The Project view has buttons for quick access to commonly used commands and options. See [Buttons and Options, on page 92](#) for details.

Tcl Commands

You can enter the Tcl (Tool Command Language) commands directly in the Tcl window, or include them in Tcl scripts that you can run in batch mode. For more information about Tcl commands, see [Tcl Commands, on page 13](#).

Document Set

This document is part of a series of books included with the Synopsys FPGA synthesis software tools. The set consists of the following books that are packaged with the tool:

- *Synopsys Synplify Pro for Microsemi User Guide*
- *Synopsys Synplify Pro for Microsemi Reference Manual*
- *Synopsys Synplify Pro for Microsemi Command Reference Manual*
- *Synopsys Synplify Pro for Microsemi Attribute Reference Manual*
- *Synopsys Synplify Pro for Microsemi Language Support Reference Manual*
- *Identify Instrumentor User Guide*
- *Identify Debugger User Guide*
- *Identify Debugging Environment Reference Manual*

CHAPTER 2

Tcl Commands

This chapter describes supported Tcl commands. The Tcl commands appear in alphabetical order.

- [Alphabetical List of Commands, on page 14](#)
- [Tcl Command Categories, on page 100](#)

Alphabetical List of Commands

The commands are listed in alphabetical order. The find, expand, and collection commands appear in the table, but are described in [Tcl Find, Expand, and Collection Commands](#), on page 101.

[add_file](#)[add_folder](#)[analyst](#)[add_to_collection](#)[all_clocks](#)[all_fanin](#)[all_fanout](#)[all_inputs](#)[all_outputs](#)[all_registers](#)[append_to_collection](#)[c_diff](#)[c_info](#)[c_intersect](#)[c_list](#)[c_print](#)[c_syndiff](#)[c_union](#)[check_fdc_query](#)[command_history](#)[constraint_file](#)[copy_collection](#)

[create_fdc_template](#)

[define_collection](#)

[define_scope_collection](#)

[design](#)

[encryptIP](#)

[encryptP1735](#)

[expand](#)

[find \(Tcl find\)](#)

[foreach_in_collection](#)

[get_cells](#)

[get_clocks](#)

[get_clock_source](#)

[get_env](#)

[get_nets](#)

[get_object_name](#)

[get_option](#)

[get_pins](#)

[get_ports](#)

[get_prop](#)

[hdl_define](#)

[hdl_param](#)

[impl](#)

[index_collection](#)

[job](#)

[log_filter](#)

[log_report](#)

[message_override](#)

[object_list](#)

[open_design](#)

[open_file](#)

[partdata](#)

[program_terminate](#)

[program_version](#)

[project](#)

[project_data](#)

[project_file](#)

[project_folder](#)

[recording](#)

[remove_from_collection](#)

[report_clocks](#)

[report_timing](#)

[run_tcl](#)

[sdc2fdc](#)

[set](#)

[set_option](#)

[sizeof_collection](#)

[status_report](#)

[synplify_pro](#)

add_file

The `add_file` command adds one or more files to a project.

Syntax

```

add_file [-filetype] fileName [ fileName [ ... ] ]
add_file -verilog fileName [ fileName [ ... ] ] [-folder folderName]
add_file -vhdl [-lib libName[ libName ] ] fileName [ fileName [ ... ] ] [-folder folderName]
add_file -include fileName [ fileName [ ... ] ]
add_file [-filetype] -job_owner par | simulation [ fileName [ ... ] ]
add_file -structver [ fileName [ ... ] ]
add_file -vlog_std standard fileName [ fileName [ ... ] ]

```

<i>-filetype</i>	Specifies the type of file being added to the project (files are placed in folders according to their file types; including this argument overrides automatic filename-extension placement). See Filename Extensions, on page 19 for a list of the recognized file types.
<i>fileName</i>	Specifies the name of the file being added to the project. Files are added to the individual project folders according to their filename extensions (View Project Files in Folders must be set in the Project View Options dialog box). You can add multiple files by separating individual filenames with a space, and you can specify different file types (extensions) within the same command.
<i>-verilog</i> or <i>-vhdl</i>	<p>Adds HDL files with non-standard extensions to the Verilog or VHDL directory, so that they can be compiled with the project. For example, the following command adds the file <code>alu.v.new</code> to the project's <code>verilog</code> directory:</p> <pre>% add_file -verilog /designs/megachip/alu.v.new</pre> <p>If you do not specify <code>-verilog</code>, the file is added to the Other directory (<code>new</code> is not a recognized Verilog extension), and the file would not be compiled with the files in the Verilog directory.</p>

[-lib <i>libName</i>]	<p>Specifies the library associated with VHDL files. The default library is work. The -lib option sets the VHDL library to <i>libName</i>.</p> <p>Note: You can also specify multiple libraries for Verilog or VHDL files.</p> <p>Verilog Example:</p> <pre>add_file -verilog -lib top -vlog_std sysv "top.v"</pre> <p>VHDL Example:</p> <pre>add_file -vhdl -lib {mylib,work} "ff.vhd"</pre> <p>Both the logical and physical libraries must be specified in the Project file (if you only specify the logical library associated with the Verilog or VHDL files, the compiler treats the module as a black box).</p>
[-folder <i>folderName</i>]	<p>Creates logical folders with custom files in various hierarchy groupings within your Project view. For example:</p> <pre>add_file -verilog -folder memory "ram_1.v" add_file -verilog -folder memory "C:/examples/verilog/common_rtl/memory/ram_1.v"</pre>
-include	<p>Indicates that the specified file is to be added to the project as an include file (include files are added to the Include directory regardless of their extension). Include files are not passed to the compiler, but are assumed to be referenced from within the HDL source code. Adding an include file to a project, although not required, allows it to be accessed in the user interface where it can be viewed, edited, or cross-probed.</p>

<code>-job_owner</code>	Allows you to determine how files are used; you can specify these options from the File Options dialog box. For example, you can automatically decide to pass files to the backend place-and-route tool (Use for Place and Route) or use them for test benches containing HDL constructs for simulation (Use for Simulation only).
<code>-structver <i>fileName</i></code>	<p>Adds structural Verilog files as input for your design project. The software performs fast compilation of the structural Verilog files, providing runtime improvements for the design. For example:</p> <pre>add_file -structver <i>fileName</i>.vm</pre> <p>For more information, see Using the Structural Verilog Flow, on page 48.</p>
<code>-vlog_std <i>standard</i></code>	<p>Overrides the global Verilog standard for an individual file. The accepted values for <i>standard</i> are v95 (Verilog 95), v2001 (Verilog 2001), and sysv (SystemVerilog). The file (<i>fileName</i>) is added to the Verilog folder in the project; the specified standard is listed after the filename in the project view and is enclosed in angle brackets (for example, <code>commchip.v <sysv></code>). Note that when you add a SystemVerilog file (a file with an <code>sv</code> extension) to a project, the <code>add_file</code> entry in the project file includes the <code>-vlog_std <i>standard</i></code> string.</p> <p>The default standard for new projects is SystemVerilog. For Verilog 2005 extensions, use <code>sysv</code> (SystemVerilog).</p>

Filename Extensions

Files with the following extensions are automatically added to their corresponding project directories; files with any other extension are added to the Other directory. The `-filetype` argument overrides automatic filename extension placement.

Extension	-Filetype	Project Folder
<code>.adc</code>	<code>-analysis_constraint</code>	Analysis Design Constraint
<code>.edf, .edn</code>	<code>-edif</code>	EDIF
<code>.fdc</code>	<code>-fpga_constraint/-constraint</code>	Logic Constraints (FDC)
<code>.sdc</code>	<code>-constraint</code>	Logic Constraints (SDC)
<code>.sv¹</code>	<code>-verilog</code>	Verilog

Extension	-Filetype	Project Folder
.tcl	-tcl	Tcl Script
.v	-verilog	Verilog
.vhd, .vhdl	-vhdl	VHDL
.vm	-structver	Structural Verilog File
any	-include	Include

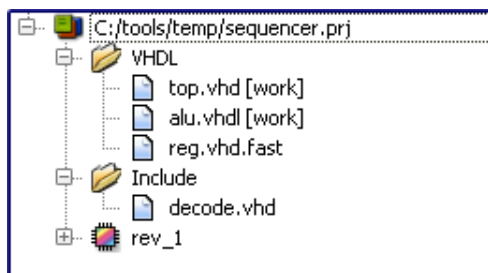
1. Use the sv format for SystemVerilog keyword support. Both Verilog and SystemVerilog formats are added to the Verilog folder.

Example: Add Files

Add a series of VHDL files to the VHDL directory and add an include file to the project:

```
% add_file /designs/sequencer/top.vhd
% add_file /designs/sequencer/alu.vhdl
% add_file -vhdl /designs/sequencer/reg.vhd.fast
% add_file -include /designs/std/decode.vhd
```

The corresponding directory structure in the Project view is shown in the following figure:



Example: File Options Designation

Designate some IP core wrappers as well as their associated instantiated component files that must be passed on to the place-and-route tool, since they are not written to the final netlist:

```
add_file -verilog -job_owner par "my_ip_core.v"
add_file -verilog -job_owner par "my_ip_core_enc.v"
```


add_folder

The `add_folder` command adds a custom folder to a project.

Syntax

add_folder *folderName*

Creates logical folders with files in various custom hierarchy groupings within your Project view. These custom folders can be specified with any name or hierarchy level.

```
add_folder verilog
```

```
add_folder verilog/common_rtl
```

```
add_folder verilog/common_rtl/prep
```

For more information about custom folders, see [Managing Project File Hierarchy](#), on page 64 in the *User Guide*.

analyst

Beta

Changes and manipulates the schematic views of the netlist.

[analyst clone_view](#) [analyst critical_path](#) [analyst dissolve](#)

[analyst filter](#) [analyst flatten](#) [analyst group](#)

[analyst pop](#) [analyst push](#) [analyst select](#)

[analyst unfilter](#) [analyst view](#)

analyst clone_view

Opens a copy of the current view.

analyst clone_view *designID*

designID

The design ID to clone. If not specified, clones the current view.

analyst critical_path

Filters the view to show the instances that are part of the critical path, if available.

analyst critical_path

analyst dissolve

Removes targeted hierarchies from the design. Contents of the hierarchy are put into the level that originally contained the hierarchy.

analyst dissolve *collection*

collection

Collection of instances to dissolve.

analyst filter

Filters the view by selected instances and ports.

analyst filter

analyst flatten

Flattens the current view.

analyst flatten

analyst group

Creates a graphical group of instances.

analyst group *collection*

collection

Instances to group. All instances must be on the same level of the hierarchy.

analyst pop

Pops up the hierarchy.

analyst pop

analyst push

Pushes down the hierarchy.

analyst push *hierarchyName*

hierarchyName

The name of the group or instance to push the hierarchy down into.

analyst select

Selects specified objects.

analyst select [*collection*] [-append] [-clear] [-instances]

collection

The ID of the collection to select.

-append

Appends objects to the selection list.

-clear

Clears the selection list.

-instances

Selects all instances in the current view.

analyst unfilter

Unfilters the view.

analyst unfilter

analyst view

Opens a schematic view.

analyst view *designID*

designID

The design ID to view.

check_fdc_query

Runs the constraint checker for constraints using the `get_*` and/or `all_*` query commands specified in the timing constraint file for the project.

Syntax

```
check_fdc_query [-full_check]
```

Arguments and Options

-full_check

Runs the full constraint checker before checking the query commands. The default is to run the `check_fdc_query` command without this option.

When the `-full_check` option is *not* specified, the command only runs the constraint syntax checker, which reduces runtime significantly, since most objects being searched are found in pre-mapping and do not require full mapping to be run. However, this option does not find bit-blasted registers and objects using the advanced `-filter @property =~` commands, where the property is created or applied during mapping because it requires optimizations such as register replication.

For example, if a 4-bit RAM output is targeted with the `get_cell` command, the differences in the results are shown below:

Command	Run Stage	Results
Default (without <code>-full_check</code>)	Pre-mapping	ram_out [3:0]
With <code>-full_check</code>	Mapping	ram_out [3] ram_out [2] ram_out [1] ram_out [0]

Description

The `check_fdc_query` command reads the `fdc` constraint file of the current project file. It runs the constraint checker for the following object query commands that are used with FDC constraints:

all_* Commands	get_* Commands
<code>all_clocks</code>	<code>get_cells</code>
<code>all_fanin</code>	<code>get_clocks</code>
<code>all_fanout</code>	<code>get_nets</code>
<code>all_inputs</code>	<code>get_pins</code>
<code>all_outputs</code>	<code>get_ports</code>
<code>all_registers</code>	

The report provides feedback on how these query commands are applied and ensures that the commands are used properly with constraints in the constraint file.

Collections created with `define_scope_collection`, `find`, and `expand` are not covered by this Tcl command. You can check these SCOPE collections in the HDL Analyst and the SCOPE interface. The report does not cover the `define_io_standard` constraint either.

Example

Invoke `check_fdc_query` from the Tcl command line for the project. You can also invoke it from a shell window.

The command writes out the results of the object query commands to the `projectName_cck_fdc.rpt` file that opens in the GUI. You may need to run the constraint checker (Run->Constraint Check) to find additional issues with constraints.

The following example shows the results of running the constraint checker in the *projectName_cck_fdc.rpt* file.

The syntax checker reports the object query commands and any issues it found and writes them to the *projectName_scck.rpt* file.

See Also

- [Constraint Checking, on page 146](#)
- [Constraint Checking Report, on page 268](#)

command_history

Displays a list of the Tcl commands executed during the current session.

Syntax

```
command_history [-save filename]
```

Arguments and Options

-save

Writes the list of Tcl commands to the specified *filename*.

Description

The `command_history` command displays a list of the Tcl commands executed during the current session. Including the `-save` option, saves the commands to the specified file to create Tcl scripts.

Examples

```
command_history -save C:/DesignsII/tut/proto/myTclScript.tcl
```

See [recording, on page 63](#).

constraint_file

The `constraint_file` command manipulates the constraint files used by the active implementation.

Syntax

```

constraint_file
  -enable constraintFileName
  -disable constraintFileName
  -list
  -all
  -clear

```

The following table describes the command arguments.

Option	Description
-enable	Selects the specified constraint file to use for the active implementation.
-disable	Excludes the specified constraint file from being used for the active implementation
-list	Lists the constraint files used by the active implementation
-all	Selects (includes) all the project constraint files for the active implementation.
-clear	Clears (excludes) all the constraint files for the active implementation

Examples

List all constraint files added to a project, then disable one of these files for the next synthesis run.

```

% constraint_file -list
attributes.fdc clocks1.fdc clocks2.fdc eight_bit_uc.fdc
% constraint_file -disable eight_bit_uc.fdc

```

Disable all constraint files previously enabled for the project, then enable only one of them for the next synthesis run.

```

% constraint_file -clear
% constraint_file -enable clocks2.fdc

```

create_fdc_template

Lets you create an initial constraint file (fdc) for your specific design.

Syntax

create_fdc_template [-period *float*] [-in_delay *float*] [-out_delay *float*]

The following table describes the `create_fdc_template` command options.

Option	Description
-period <i>float</i>	Specifies the default values for port clocks.
-in_delay <i>float</i>	Specifies the default values for the input delay ports.
-out_delay <i>float</i>	Specifies the default values for the output delay ports.

design

Beta

Returns netlist data representing information about the design. Commands are available in both batch and GUI mode.

design c_diff	design c_filter	design c_intersect
design c_list	design c_print	design c_union
design close	design expand	design find
design get_prop	design get	design list
design open	design set	

design c_diff

Returns a new find collection containing the differences between two existing find collections.

Syntax

design c_diff *collection1 collection2*

collection1

The first collection to compare.

collection2

The second collection to compare.

design c_filter

Filters a find collection based on set properties.

Syntax

design c_filter *collection pattern* [-inst] [-net] [-port] [-pin]

collection

The collection ID to filter.

pattern

Statement used to filter.

-inst

Returns matching instances. If no -type option (-inst, -net, -port, or -pin) is set, all types will be returned.

-net

Returns matching nets. If no -type option (-inst, -net, -port, or -pin) is set, all types will be returned.

-port

Returns matching ports. If no -type option (-inst, -net, -port, or -pin) is set, all types will be returned.

-pin

Returns matching pins. If no -type option (-inst, -net, -port, or -pin) is set, all types will be returned.

design c_intersect

Defines common objects that are included in each of the collections being compared.

Syntax

design c_intersect *collectionList*

collectionList

List of collections separated by spaces.

design c_list

Converts a collection to a Tcl list of objects.

Syntax

design c_list *collection*

collection

Collection to convert.

design c_print

Displays collections or properties in column format.

Syntax

design c_print *collection* [-**prop** *propertyName*] [-**file** *filename*]

collection

The collection to print as a table.

-prop

Writes a column in the table for properties of type *propname*.

-file

Writes the collection to *filename*.

design c_union

Combines multiple collections into a single collection.

Syntax

design c_union *collectionList*

collectionList

Space-separated list of collections.

design close

Closes the specified design ID. If no design ID is provided, this command closes the current active design.

Syntax

design close *designID*

designID

The design ID to close.

design expand

The design expand command identifies objects based on their connectivity, by expanding forward from a given starting point.

Syntax

design expand [-*objectType*] [-from *object*] [-thru *object*] [-to *object*] [-level *integer*]
[-hier] [-leaf] [-seq] [-print]

-objectType

Optionally specifies the type of object to be returned by the expansion. If you do not specify *objectType*, all objects are returned. The object type is one of the following:

-inst – returns all instances between the expansion points. This is the default.

-pin – returns all instance pins between the expansion points.

-net – returns all nets between the expansion points.

-port – returns all top-level ports between the expansion points.

-from *object*

Specifies a list or collection of ports, instances, pins, or nets for expansion forward from all listed pins. Instances and input pins are automatically expanded to all output pins of the instances. Nets are expanded to all output pins connected to the net. If you do not specify this argument, backward propagation stops at a sequential element.

-thru *object*

Specifies a list or collection of instances, pins, or nets for expansion forward or backward from all listed output pins and input pins respectively. Instances are automatically expanded to all input/output pins of the instances. Nets are expanded to all input/output pins connected to the net. You can have multiple -thru lists for product of sum (POS) operations.

-to *object*

Specifies a list or collection of ports, instances, pins, or nets for expansion backward from all the pins listed. Instances and output pins are

automatically expanded to all input pins of the instances. Nets are expanded to all input pins connected to the net. If you do not specify this argument, forward propagation stops at a sequential element.

-level *integer*

Limits the expansion to N logic levels of propagation. You cannot specify more than one -from, -thru, or -to point when using this option.

-hier

Modifies the range of any expansion to any level below the current view. The default for the current view is the top level and is defined with the `define_current_design` command as in the compile-point flow.

-leaf

Returns only non-hierarchical instances.

-seq

Modifies the range of any expansion to include only sequential elements. By default, the `expand` command returns all object types. If you want just sequential instances, make sure to define the *object_type* with the -inst argument, so that you limit the command to just instances.

-print

Evaluates the `expand` function and prints the first 20 results. If you use this command from HDL Analyst, results are printed to the Tcl window; for constraint-file commands, the results are printed to the log file at the start of the Mapper section. For a full list of objects found, you must use `c_print` or `c_list`. Reported object names have prefixes that identify the object type. There are curly braces around each name to allow for spaces in the names. For example:

```
{i:reg1}  
{i:reg2}  
{i:\weird_name[foo$]}  
{i:reg3}  
<<found 233 objects. Displaying first 20 objects. Use  
  c_print or c_list for all. >>
```

design find

Identifies design objects based on specified criteria.

Syntax

design find

[-objectType] *pattern*
[-seq]
[-hier [-hsc character]]
[-inst instance]
[-net net]
[-port port]
[-pin pin]
[-view view]
[-depth viewNumber]
[-flat]
[-print]
[-filter expression]

-objectType *pattern*

Specifies the type of object to be found. Object types are view, inst, port, pin, or net. The *pattern* argument is required and specifies the search pattern to be matched. The pattern can include the * and ? wildcard characters (see [Regular Expressions, Wildcards, and Special Characters, on page 214](#)).

-seq

Finds sequential (clocked) instances (the -inst object type is not required). This argument is equivalent to -filter @is_sequential.

-hier

Extends the search downward through each level of the local hierarchy, instead of limiting the search to the current view. The default hierarchy separator for the search is the period (.).

-inst *instance*

Finds instances. If no -type option is set, find defaults to finding instances, nets, and ports.

-net *net*

Finds nets. If no -type option is set, find defaults to finding instances, nets, and ports.

-port *port*

Finds ports. If no -type option is set, find defaults to finding instances, nets, and ports.

-pin *pin*

Finds pins. If no -type option is set, find defaults to finding instances, nets, and ports.

-view *view*

Finds views. If no **-type** option is set, find defaults to finding instances, nets, and ports.

-depth *depth*

Sets the start depth for the search. *depth* may be a single hierarchy depth or a range. Using **-depth** with a range will cause **-hier** and **-flat** arguments to be ignored.

-flat

Extends the search to all levels, but with **-flat**, the * wildcard character matches hierarchy separators as well as characters. This means that the following example finds instance `a1_fft` at the current level as well as the hierarchical instance `a1.fft`:

```
find -seq -flat a1*fft
```

-print

Prints the first 20 search results. For a full list of objects found, use `c_print` or `c_list`. If you use `find` from the shell, the results are printed to the Tcl window; if you find in the constraint file, the results are printed to the log file at the beginning of the Mapper section. Reported object names have prefixes that identify the object type and curly braces around each name to allow for spaces in the names as shown below:

```
{i:reg1}
{i:\weird_name [foo$]}
{i:reg2}
<<found 233 objects. Displaying first 20 objects. Use c_print
or c_list for all. >>
```

-filter *expression*

Further refines the results of `find` by filtering the results using the specified object property. For syntax details, refer to [find -filter, on page 224](#).

design get_prop

Returns a list of property values for an object or collection.

Syntax

```
design get_prop [objectName | collection] [-prop value]
```

objectName | *collection*

The object or collection to use.

-prop

The property value to return.

design get

Returns the design ID for the current active design.

Syntax

design get

design list

Returns a list of available design IDs.

Syntax

design list

design open

View schematic of the design in its current state.

Syntax

design open [-list] [-verbose] [-database *value*] [*netlist*]

-list

Lists netlists available for viewing in the current state

-verbose

Use with **-list**. Prints short description of each available netlist.

-database *value*

Path to the database if different than the current database.

netlist

The netlist to view.

design set

Sets specified design ID as the active design.

Syntax

design set *designID*

designID

The design ID to set as active.

encryptIP

Runs a Perl script that lets IP vendors provide encrypted evaluation IP to synthesis users. The IP is encrypted using the OpenIp scheme. You can download the encryptIP Perl script from SolvNet. See the article published at <https://solvnet.synopsys.com/retrieve/032343.html>.

For additional information about the script, see [The encryptIP Script](#), on [page 364](#) in the *Reference* manual.

Syntax

encryptIP

-in | **input** *inputFile*
-out | **output** *outputFileName*
-c | **cipher** "{des-cbc | 3des-cbc | aes128-cbc}"
-k | **key** *symmetricEncryptionKeyInTextFormat*
-kx | **keyx** *symmetricEncryptionKeyInHexadecimalFormat*
-bd | **build_date** *ddmmmyyyy*
-om | **outputmethod** "{plaintext | blackbox | persistent_key}"
-incv | **includevendor** *vendorKeyBlock*
-dkn | **datakeyname** *sessionKeyName*
-dko | **datakeyowner** *sessionKeyOwner*
-a | **author** *dataAuthor*
-v | **verbose**

You must specify all required parameters.

-in input	Names the input RTL file to be encrypted.
-out output	Names the output file generated after encryption.
-c cipher	<p>Specifies the symmetric encryption cipher. The keylength must match the algorithm being used, with each character using 8 bits.</p> <ul style="list-style-type: none"> • des-cbc specifies the Data Encryption Standard (DES); uses a 64-bit key. • 3des-cbc specifies the Triple Data Encryption Standard (Triple DES); uses a 192-bit key. • aes128-cbc specifies the Advanced Encryption Standard (AES Rijndael); uses a 128-bit key. <p>See Encryption and Decryption Methodologies, on page 359 in the <i>Command Reference</i> for an overview. For information about how symmetric encryption fits into the encryption flow, see Encryption and Decryption, on page 489 in the <i>User Guide</i>.</p>
-k key	Specifies the symmetric data decryption key used to encode your RTL data block. The key is in text format, and can be any string (e.g. ABCDEFG). The exact length of the key depends on the data method you use. In a future release, this key will be automatically generated if you do not specify one.
-kx keyx*	Optional parameter. Specifies the symmetric encryption key in hexadecimal format.
-bd build_date	Specifies a date (ddmmmyyyy). The IP only works in Synplicity software released after the specified date. It is recommended that you use a date in January 2008 or later. For example:16FEB2008. This option lets you force users to use newer Synopsys FPGA releases that contain more security features. Contact Synopsys if you need help in deciding what build date to use.
-om outputmethod	<p>Determines how the IP is treated in the output after synthesis:</p> <ul style="list-style-type: none"> • plaintext specifies that the IP is unencrypted in the synthesis netlist. • blackbox specifies that the IP is treated as a black box, and only interface information is in the output. • persistent_key is the default setting. <p>See Output Methods for encryptIP, on page 38 for more information.</p>
-incv includevendor	Optional parameter that specifies a key block for an EDA vendor, so that IP can be read by the vendor tools. C

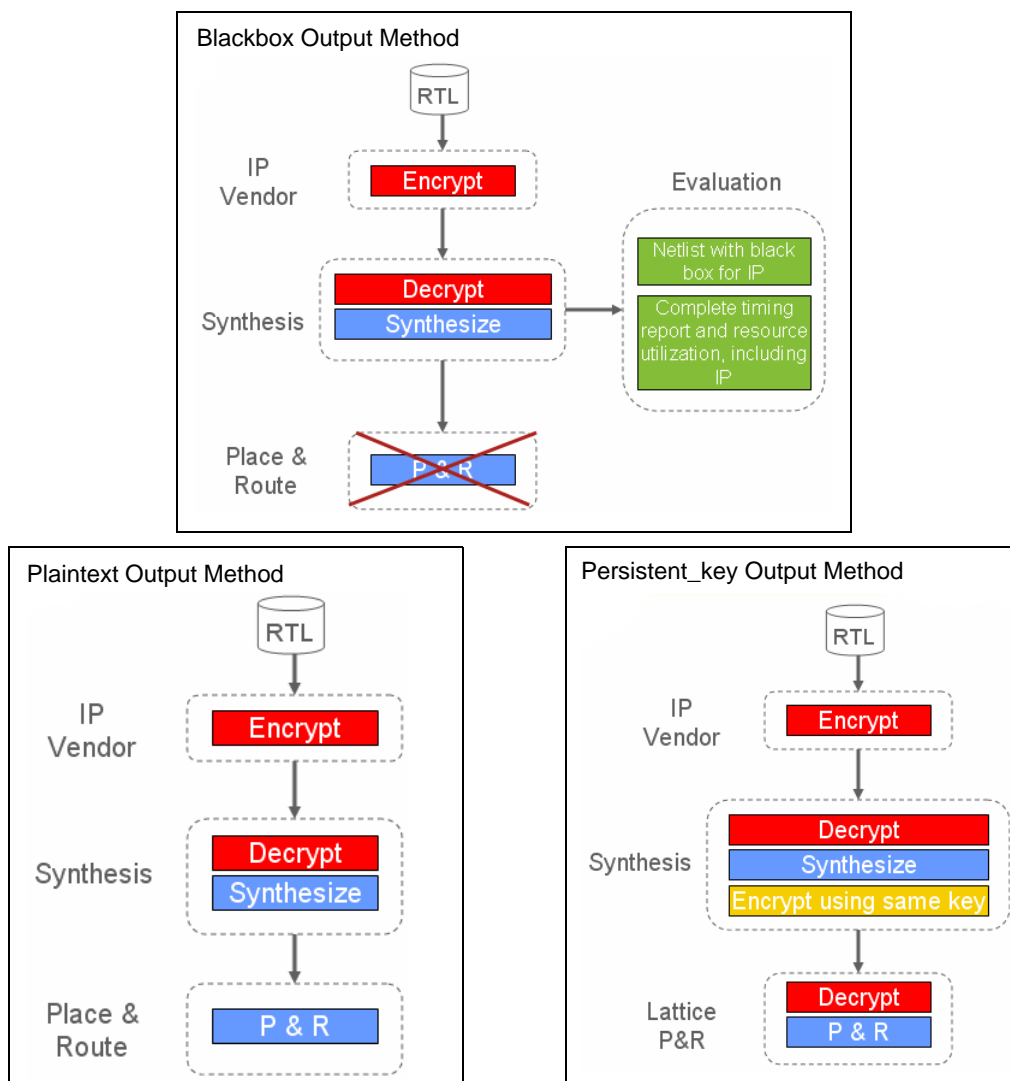
-dkn datakeyname	Specifies a string that denotes your session key, that was used to encrypt your IP.
-dko datakeyowner	Optional parameter that names the owner of the session key. The value can be any string.
-a author	Optional parameter that names the author of the session key. The value can be any string.
-v verbose	Specifies that the script run in verbose mode.

Output Methods for encryptIP

You can control the level of IP protection in the synthesis output netlist by specifying an output method when you encrypt your IP by running the encryptIP script. For example, -om blackbox. The output method is included in the encrypted key block of your encrypted RTL. The table below shows the values for -om. See [Specifying the Script Output Method, on page 501](#) in the *User Guide* for guidelines on how to use these methods effectively.

Output Method	Description
blackbox	With this method, you cannot run gate-level simulation on the output netlist or place and route the IP. This is because the output netlist contains the IP interface only and no IP contents. It only includes ports and connections; there are no nets or instances shown inside the IP.
plaintext	With this method, you can synthesize, run gate-level simulation, place, route, and implement an FPGA on the board that includes your IP. The output netlist contains your unencrypted IP, which is completely readable.
persistent_key	By default, the output is encrypted using the same session key and cipher you used to encrypt your IP for the synthesis tools.

The following figure shows how the synthesis tools work with each of the three encryptIP output methods:



Effect of Output Method on Viewing IP

In the synthesis tools, the contents of the IP are always shown as black boxes, and you cannot view the contents. In the Technology view, you cannot view the initialization values for the LUT.

Effect of Output Method on Output Constraints

After synthesis, the output constraints generated for the IP are not encrypted, regardless of the output method. They are always readable.

Effect of Output Method on Output Netlist

The following table summarizes how different output methods affect the output:

Method (-om)	Output Netlist After Synthesis
blackbox	The output netlist contains the IP interface only and no IP content, and only includes IP ports and connections. The IPs are treated as black boxes, and there are no nets or instances shown inside the IP. This content applies to all netlist formats generated for different vendors, whether it is HDL (vnm or vhm), EDIF (edf or edn).
plaintext	The output netlist contains your unencrypted IP, which is completely readable (nothing is encrypted).
persistent_key	The output netlist includes encrypted versions of the IP. Specifics differ, based on the target.

encryptP1735

Runs a script that allows IP vendors to encrypt modules or components that can then be downloaded for evaluation or used by a Synopsys FPGA user.

The encryptP1735 script supports three different use models for encrypting RTL files. See [The encryptP1735 Script, on page 360](#) in the *Reference* manual for more information on the use models and the files they use.

Syntax

encryptP1735

```
-l | list listofFiles
[-pk | public_keys keyFileName]
[-sk | showkey]
[-verbose]
[-verilog]
[-vhdl]
[-log logFileName]
[-h | -help]
```

The following table describes the command-line arguments.

-l list	Specifies a list of the files to be encrypted; <i>listofFiles</i> is a list of the non-encrypted HDL input files with each filename entry on a separate line.
-pk public_keys	Specifies the public keys repository file. This file contains public keys for various tools. If the encryption envelope contains a key block with a particular keyowner and keyname, the public keys file is searched by the script to find a public key to use during key-block generation. See Public Keys Repository File, on page 361 in the <i>Reference</i> manual for information about this file.
-sk showkey	When used, the encryption script displays the session key in use (useful when random keys are used and the user wants to make a note of the key being used).
-verbose	Prints more detailed messages to the screen or log file.
-verilog	Specifies Verilog HDL file format when filename does not include a .v or .sv extension.
-vhdl	Specifies VHDL HDL file format when filename does not include a .vhd or .vhdl extension.
-log	Prints messages to the specified log file.

Examples

The example below encrypts the files in mylist and uses the default keys.txt file as the public keys file. Resulting messages are written to the encrypt.log file.

```
perl encryptP1735.pl -l mylist -log encrypt.log
```

The following example illustrates a similar command using an explicit public-keys file:

```
perl encryptP1735.pl -l mylist -pk public_keys.txt  
-log encrypt.log
```

get_env

The `get_env` command reports the value of a predefined system variable.

Syntax

```
get_env systemVariable
```

Use this command to view system variable values. The following example shows you how to use the `get_env` command to see the value of the previously created `MY_PROJECT` environment variable. The `MY_PROJECT` variable contains the path to an HDL file directory, so `get_env` reports this path.

```
get_env MY_PROJECT  
d:\project\hdl_files
```

In the project file or a Tcl script, you can define a Tcl variable that contains the environment variable. In this example, `my_project_dir` contains the `MY_PROJECT` variable, which points to an HDL file directory.

```
set my_project_dir [get_env MY_PROJECT]
```

Then, use the `$systemVariable` syntax to access the variable value. This is useful for specifying paths in your scripts, as in the following example which adds the file `myfile1.v` to the project.

```
add_file $my_project_dir/myfile1.v
```

get_option

The `get_option` command reports the settings of predefined project and device options. The options are the same as those for `set_option`. See [set_option](#), on [page 66](#) for details.

Syntax

```
get_option -optionName
```

hdl_define

For Verilog designs, this command specifies values for Verilog text macros. You can specify text macro values that you would normally enter using the Verilog ``define` statement in a Verilog file included at the top of the synthesis project. The parameter value is valid for the current implementation only.

This command is equivalent to the `set_option -hdl_define` command.

Syntax

```
hdl_define
  -set "directive=value [directive=value ...]"
  -clear
  -list
```

Examples

```
hdl_define -set "SIZE=32"
```

This statement specifies the value 32 for the `SIZE` directive; the following statement is written to the project file:

```
set_option -hdl_define -set "SIZE=32"
```

To define multiple directive values using `hdl_define`, enclose the directives in quotes and use a space delimiter. For example:

```
hdl_define -set "SIZE=32 WIDTH=8"
```

The software writes the following statement to the `prj` file:

```
set_option -hdl_define -set "size=32 width=8"
```

See Also

[Compiler Directives and Design Parameters, on page 231](#) for information on specifying compiler directives in the GUI.

hdl_param

The `hdl_param` command shows or sets HDL parameter overrides. For the GUI equivalent of this command, select Project->Implementation Options->Verilog/VHDL.

Syntax

```
hdl_param
  -add {paramName}
  -list | -set paramName {paramValue}
  -clear
  -overrides
```

The following table describes the command arguments.

Option	Description
-add	Adds a parameter override to the project.
-list	Shows parameters for the top-level module only and lists values for parameters if there is a parameter override.
-set	Sets a parameter override and its value for the active implementation. Only the parameter value is enclosed within curly braces.
-clear	Clears all parameter overrides of the active implementation.
-overrides	Lists all the parameter override values used in this project.

Examples

In batch mode, to set generic values using the `set_option` command in a project file, specify the `hdl_param` generic with quotes and enclose it within `{}`. For example:

```
set_option -hdl_param -set ram_file {"init.mem"}
set_option -hdl_param -set simulation {"false"}
```

Suppose the following parameter is set for the top-level module.

```
set_option -hdl_param -set {"width=8"}
```


Add a parameter override and its value, then list the parameter override.

```
hdl_param -add {"size=32"}
hdl_param -list "size=32"
```

impl

The impl command adds, removes, or modifies an implementation.

Syntax

```
impl
  -add [implName] [model]
  -name implName
  -remove implName
  -active [implName]
  -list
  -type implType
  -result_file
  -dir
```

The following table describes the command arguments.

Option	Description
-add	Adds a new device implementation. If: <ul style="list-style-type: none"> • <i>implName</i> is not specified, creates a unique implementation name by incrementing the name of the active implementation. • you want to add a new implementation copied from implementation <i>model</i>.
-name	Changes the name of the active implementation.
-remove	Removes the specified implementation.
-active	Reports the active implementation. If you specify an implementation name, changes the specified name to the active implementation.
-list	Lists all the implementations used in this project.

Option	Description
-type	Specifies the type of implementation to add. For example, the: <ul style="list-style-type: none"> • -type fpga option creates an FPGA implementation. • -type identify option creates an Identify implementation.
-result_file	Displays the implementation results file.
-dir	Displays the implementation directory.

Examples

The following command sequence lists all implementations, reports the active implementation, and then activates a different implementation.

```
% impl -list
design_worst design_typical design_best

% impl -active
design_best

%impl -active design_typical

% impl -active
design_typical

% impl -add rev_1_identify mixed -type identify
```

job

The job command, for place and route job support, creates, removes, identifies, runs, cancels, and sets/gets options for named P&R jobs.

Syntax

```
job jobName [-add jobType |-remove |-type |-run [mode] |-cancel |
               -option optionName [optionValue] ]
```

```
job -list
```

The following table describes the command options.

Option	Description
-run	Runs the P&R job, according to the specified options:
-add <i>jobType</i>	Creates a new P&R job for the active implementation.
-cancel	Cancels a P&R job in progress.
-remove	Removes a P&R job from an active implementation
-list	Returns a list of the P&R jobs in the active implementation.
-remove	Removes a P&R job from the active implementation
-option <i>optionName</i> [<i>optionValue</i>]	Get/set options for <i>jobName</i> .
-type	Returns the P&R job type.

Examples

```
% job pr_2 -add par
% job pr_2 -option enable_run 1
% job pr_2 -option run_backannotation 1
% job pr_2 -run
```

log_filter

This command lets you filter errors, notes, and warning messages. The GUI equivalent of this command is the Warning Filter dialog box, which you access by selecting the Warnings tab in the Tcl window and then clicking Filter. For information about using this command, see [Filtering Messages in the Message Viewer, on page 199](#) in the *User Guide*.

Syntax

```
log_filter -field fieldName==value
log_filter -show_matches
log_filter -hide_matches
log_filter -enable
log_filter -disable
log_filter -clear
```

The following table shows valid *fieldName* and *value* values for the `-field` option:

Fieldname	Value
type	Error Warning Note
id	The message ID number. For example, MF138
message	The text of the message. You can use wildcards.
source_loc	The name of the HDL file that generated the message.
log_loc	The corresponding srr file (log).
time	The time the message was generated.
report	The log file section. For example, Compiler or Mapper.

Example

```
log_filter -hide_matches
log_filter -field type==Warning -field message==*Una*
        -field source_loc==sendpacket.v -field log_loc==usbHostSlave.srr
        -field report=="Compiler Report"
log_filter -field type==Note
log_filter -field id==BN132
log_filter -field id==CL169
log_filter -field message=="Input *"
log_filter -field report=="Compiler Report"
```

log_report

This command lets you write out the results of the `log_filter` command to a file. For information about using this command, see [Filtering Messages in the Message Viewer, on page 199](#) in the *User Guide*.

Syntax

You specify this command after the `log_filter` commands.

log_report -print *fileName*

Example

```
log_report -print output.txt
```

message_override

Allows you to suppress or override the log file message ID specifications.

Syntax

```
message_override [ -suppress value ] [ -read_file value ] [ -error value ]  
[ -warning value ] [ -note value ] [ -remove value ] [ -global ] [ -clear ]
```

The following table describes the command arguments and options.

Option	Description
-suppress <i>value</i>	Lists message IDs to suppress in the log file.
-read_file <i>value</i>	Reads the specified message override file.
-error <i>value</i>	Lists the message ID type as an error.
-warning <i>value</i>	Lists the message ID type as a warning.
-note <i>value</i>	Lists the message ID type as a note.
-remove <i>value</i>	Removes the override and resets the message type to its original value.
-global	Allows message operations to be applied globally. Otherwise, the override operation is only applied on messages for the current project.
-clear	Removes all overrides and resets messages to their original types.

For example:

```
message_override -error MF446
```

treats this message as an error.

open_design

The `open_design` command specifies a netlist file (`srs`, `srp`, or `srm`) that can be used to search the database with the Tcl `find` command in batch mode. With `open_design`, you can use `find` without having to open an RTL, partitioned RTL, or Technology view. Use `open_design` to read in the `srs`, `srp`, or `srm` file before issuing the `find` command. See the example below.

Syntax

`open_design filename`

Where:

filename is the RTL (`srs`), partitioned RTL (`srp`), or Technology (`srm`) file that can be used to search the database. The specified netlist is loaded on-demand to minimize memory resources.

Example

```
project -load ../examples/vhdl/prep2_2.prj
open_design prep2_2.srs
set a [find -inst *]
c_print $a -file a.txt
open_design prep2_2.srm
set b [find -net *]
c_print $b -file b.txt
```

In the example above, `prep2_2` is loaded and the information from the RTL view file is read in. Then, the `find` command searches for all instances in the design and prints them to file `a`. Next, the technology view file is read in, then `find` searches for all nets in the design and prints them to file `b`.

See [find](#), on page 102.

open_file

The `open_file` command opens views within the tool. The command accepts two arguments: `-rtl_view` and `-technology_view`.

Syntax

`open_file -rtl_view |-technology_view`

The `-rtl_view` option displays the RTL view for the current implementation, and the `-technology_view` option displays the technology view for the current implementation. Views remain displayed until overwritten and multiple views can be displayed.

partdata

The `partdata` command loads part files and returns information regarding a part such as available families, family parts, vendors, attributes, grades, packages.

Syntax

```
partdata
  -load filename
  -family
  -part family
  -vendor family
  -attribute attribute family
  -grade [family:]part
  -package [family:]part
  -oem [family:]part
```

Option	Description
-load <i>filename</i>	Loads part file.
-family	Lists available technology families.
-part <i>family</i>	Lists all parts in specified family.
-vendor <i>family</i>	Returns vendor name for the specified family.
-attribute <i>attribute family</i>	Returns the value of the job attribute for the specified family.
-grade [<i>family:</i>] <i>part</i>	Lists the speed grades available for the specified part.
-package [<i>family:</i>] <i>part</i>	Lists the packages available for the specified part.
-oem [<i>family:</i>] <i>part</i>	Returns true if the part entered is an OEM part.

Example

The following example prints out the available vendors, their supported families, and the parts for each family.

```
% foreach vendor [partdata -vendorlist]
% puts VENDOR:$vendor;
% foreach family [partdata -family $vendor]
% puts \tFAMILY:$family;
% puts \t\tPARTS:;
% foreach part [partdata -part $family]
% puts \t\t$part;
```

program_terminate

Immediately terminates the tool session without prompting or saving any data.

Syntax

program_terminate

Arguments and Options

None

Description

The `program_terminate` command terminates a tool session without prompting or saving data. Use this command with caution as any unsaved data is lost and cannot be recovered.

Examples

```
program_terminate
```

program_version

Returns the product and software release version.

Syntax

program_version

Arguments and Options

None

Description

The `program_version` command returns the software product version number.

Examples

```
% program_version
Synplify Pro J-2014.09
```

project

The `project` command runs job flows to create, load, save, and close projects, to change and examine project status, and to archive projects.

Syntax

```
project -run [-all] [-bg] implementationList [-impl implementationName]
[-clean] implementationList [-impl implementationName]
[-parallel] implementationList [-impl implementationName]
```

```
project {-new [projectPath] | -load projectPath | -close [projectPath]
|-save [projectPath] | -insert projectPath} |
```

```
project {-active [projectName] | -dir | -file | -name | -list | -filelist |
-fileorder filepath1 filepath2 [... filepathN] | -addfile filepath |
-movefile filepath1 [filepath2] | -removefile filepath}
```

```
project {-result_file resultFilePath | -log_file [logfileName]}
```

```
project -copy [-project filename] [-implement implementationName]
[-dest_dir pathname] [-copy_type {full | local | customize}]
[-add_srs [fileList] -no_input]
```

```
project -unarchive [-archive_file pathname/filename] [-dest_dir pathname]
```

run option

The run option lets you synthesize selected implementations of a Project file. You can choose to use the arguments for the run option independently or in any combination. The arguments available are described in the table below.

You can also use the Batch Run Setup dialog box to set the arguments to use with the run option. For details, see [Run Implementations Setup Command, on page 248](#).

Option	Description
-run [-all] [-bg] [-clean] [-parallel]	Synthesizes the project, according to the specified options: You can use <i>run</i> with any of the following arguments: <ul style="list-style-type: none">• -all – Runs all implementations of the active project.• -bg – Runs specified implementations in non-blocking background mode.• -clean – Runs specified implementations, while ignoring up-to-date checking. This option cleans all previous results and forces a complete rerun.• -parallel – Runs specified implementations concurrently. Additional licenses are required for each job.

Option	Description
	<p>The <i>mode</i> can be one of the following keywords:</p> <ul style="list-style-type: none"> • compile – Compiles the active project, but does not map it. • constraint_check - Validates the syntax and applicability of constraints defined in one or more constraint files. • fsm_explorer – Selects optimum FSM-encoding style for finite-state machines. • netlist_optimizer – Runs netlist optimization. • syntax_check – Verifies that the HDL is syntactically correct; errors are reported in the log file. • synthesis – Default mode if no mode is specified. Compiles (if necessary) and synthesizes the currently active project. If followed by the <code>-clean</code> option (<code>project -run synthesis -clean</code>), resynthesizes the entire project, including the top level and <i>all compile points</i>, whether or not their constraints, implementation options or source code changed since the last synthesis. If not followed by <code>-clean</code>, only compile points that have been modified are resynthesized. • synthesis_check – Verifies that the design is functionally correct; errors are reported in the log file. • timing – Runs the Timing Analyst. This is equivalent to clicking the Generate Timing button in the Timing Report Generation dialog box with user-specified values.
	<ul style="list-style-type: none"> • write_netlist – Writes the mapped output netlist to structural Verilog (<code>vm</code>) or VHDL (<code>vhm</code>) format. You can also use this command in an incremental timing analysis flow. For details, see Run Menu, on page 244 and Generating Custom Timing Reports with STA, on page 332.

The following table describes the rest of the Project file command options.

Option	Description
-new [<i>projectPath</i>]	Creates a new project in the current working directory. If <i>projectPath</i> is specified, creates the project in the specified directory.
-load <i>projectPath</i>	Opens and loads the project file specified by <i>projectPath</i> .
-close [<i>projectPath</i>]	Closes the currently active project. If <i>projectPath</i> is specified, closes the specified project.
-save [<i>projectPath</i>]	Saves the currently active project. If <i>projectPath</i> is specified, saves the specified project.
-insert <i>projectPath</i>	Adds the specified project to the workspace project.
-active [<i>projectName</i>]	Shows the active project. If <i>projectName</i> is specified, makes the specified project the active project.
-dir	Shows the project directory for the active project.
-file	Returns the path to the active project.
-name	Returns the filename (prj) of the active project.
-list	Returns a list of the loaded projects.
-filelist	Returns the pathnames of the files in the active project.
-fileorder <i>filepath1</i> <i>filepath2</i> [... <i>filepathN</i>]	Reorders files by adding the specified files to the end of the project file list.
-addfile <i>filepath</i>	Adds the specified file to the project.
-movefile <i>filepath1</i> [<i>filepath2</i>]	Moves <i>filepath1</i> to follow <i>filepath2</i> in project file list. If <i>filepath2</i> is not specified, moves <i>filepath1</i> to top of list.
-removefile <i>filepath</i>	Removes the specified file from the project.
-result_file <i>resultFilePath</i>	Changes the name of the synthesis result file to the path specified.
-log_file [<i>logfileName</i>]	Reports the name of the project log file. If <i>logfileName</i> is specified, changes the base name of the log file.

Option	Description
-archive	
-project <i>filename</i>	• project <i>filename</i> – copies a project other than the active project. If you do not use this option, by default the active project is copied.
[- root_dir <i>pathname</i>]	• root_dir <i>pathname</i> – specifies the top-level directory containing the project files.
-archive_file <i>filename.sar</i>	• archive_file <i>filename</i> – is the name of the archived project file.
-archive_type {full local customize}	• archive_type - specifies the type of archive:
-add_srs [<i>fileList</i>]	• full – performs a complete archive; all input and result files are contained in the archive file.
-no_input	• customize – performs a partial archive; only the project files that you select are included in the archive.
	• local – includes only project input files in the archive; does not include result files.
	• add_srs – adds the listed srs files to the archived project. Use the -no_input option with this command. If <i>fileList</i> is omitted, adds all srs files for the project/implementations. The srs files are the RTL schematic views that are output when the design is compiled (Run->Compile Only).
	For more information about, and examples of the project -archive command, see Archive Utility, on page 59 .

Option	Description
-copy -project <i>filename</i> -implement <i>implementationName</i> -dest_dir <i>pathname</i> -copy_type {full local customize} -add_srs [<i>fileList</i>] -no_input	<ul style="list-style-type: none"> • project <i>filename</i> – copies a project other than the active project. If you do not use this option, by default the active project is copied. • implement <i>implementation_name</i> – archives all files in the specified implementation. • dest_dir <i>directory_pathname</i> – specifies the directory in which to copy the project files. • copy_type – specifies the type of file/project copy: <ul style="list-style-type: none"> • full – performs a complete copy; all input and result files are contained in the archive file. • customize – performs a partial copy; only the project files that you select are included in the archive. • local – includes only project input files in the copy; does not include result files. • add_srs – adds the listed srs files to the archived project. Use the -no_input option with this command. If <i>fileList</i> is omitted, adds all srs files for the project/implementations. The srs files are the RTL schematic views that are output when the design is compiled (Run->Compile Only). <p>For more information about, and examples of the project -copy command, see Archive Utility, on page 59.</p>
-unarchive -archive_file <i>pathname/filename</i> -dest_dir <i>pathname</i>	<ul style="list-style-type: none"> • archive_file <i>pathname/filename</i> – is the name of the archived project file. • dest_dir <i>pathname</i> – specifies the directory in which to write the project files. <p>For more information about, and examples of the project -unarchive command, see Archive Utility, on page 59.</p>

project Command Examples

Load the project top.prj and compile the design without mapping it. Compiling makes it possible to create a constraint file with the SCOPE spreadsheet and display an RTL schematic representation of the design.

```
% project -load top.prj
% project -run compile
```

Load a project and synthesize the design.

```
% project -load top.prj
% project -run synthesis
```

In the example above, you can also use the command `project -run`, since the default is synthesis.

Archive Utility

The archive utility provides a way to archive, extract, or copy your design projects. An archive file is in Synopsys proprietary format and is saved to a file name using the `sar` extension. You can also use this utility to submit your design along with a request for technical support.

The archive utility is available through the Project menu in the GUI or through the `project` Tcl command. See the following for details:

For information about ...	See ...
Archiving, un-archiving, or copying projects	Archiving Files and Projects, on page 99 in the <i>User Guide</i>

Project Archive Examples

The following example archives all files in the project and stores the files in the specified `sar` file:

```
project -archive -project c:/proj1.prj
       -archive_file c:/archive/proj1.sar
```

The next example archives the project file (`prj`) and all local input files into the specified `sar` file.

```
project -archive -project c:/proj1.prj -archive_type local
       -archive_file c:/archive/proj1.sar
```

The following example archives the project file (`prj`) only for selected `srs` files into the specified `sar` file. Any input source files that are in the project are not included.

```
project -archive -project c:/proj1.prj -archive_type customize
       -add_srs -no_input -archive_file c:/archive/proj1.sar
```

Project Unarchive Example

The following example extracts the project files from `c:/archive/proj1.sar` to directory `c:/proj1`. All directories and sub-directories are created if they do not already exist.

```
project -unarchive -archive_file c:/archive/proj1.sar
        -dest_dir c:/proj1
```

Project Copy Examples

The following example copies only selected srs files for the project to the destination project file directory.

```
project -copy -project d:/test/proj_2.prj -copy_type customize
        -add_srs -no_input -dest_dir d:/test_1
```

The next example copies all input source files and srs files selected for the project to the destination project file directory.

```
project -copy -project d:/test/proj_2.prj -copy_type customize
        -dest_dir d:/test_1
```

project_data

The `project_data` command shows or sets properties of a project.

Syntax

```
project_data {-active [ projectName ] | -dir | -file }
```

The following table describes the command options.

Option	Description
-active	Set/show active project. With no argument, shows the active project. If <i>projectName</i> is specified, changes the active project to <i>projectName</i> .
-dir	Show directory of active project.
-file	Show the project file for the active project. The full path is included with the file name.

project_file

The `project_file` command manipulates and examines project files.

Syntax

```
project_file {-lib fileName [libName] | -name fileName [newPath] |  
-time fileName [format] | -date fileName | -type fileName |  
-savetype fileName [relative | absolute] -move fileName1 [fileName2] |  
-remove fileName | -top topModule |
```

The following table describes the command options.

Option	Description
-lib	Shows the project file library associated with <i>fileName</i> . If <i>libName</i> is specified, changes the project file library for the specified file to <i>libName</i> .
-name	Shows the project file path for the specified file. If <i>newPath</i> is specified, changes the location of the specified project file to the directory path specified by <i>newPath</i> .
-time	Shows the file time stamp. If a <i>format</i> is specified, changes the composition of the time stamp according to the combination of the following time formatting codes: %H (hour 00-23) %M (minute 00-59) %S (second 00-59) %d (day 01-31) %b (abbreviated month) %Y (year with century)
-date	Shows the file date.
-type	Shows the file type.
-savetype	Sets or shows whether a file is saved relative to the project or its absolute path.
-move	Positions <i>fileName1</i> after <i>fileName2</i> in HDL file list. If <i>fileName2</i> is not specified, moves <i>fileName1</i> to the top of the list.
-remove	Removes the specified file from the project file list.
-top	Sets or shows the top-level module of the specified file for the active implementation.

Examples

List the files added to a project. Remove a file.

```
% project -filelist path_name1/cpu.v path_name1/cpu_cntrl.v
    path_name2/cpu_cntrl.vhd
% project_file -remove path_name2/cpu_cntrl.vhd
```

project_folder

The `project_folder` command manipulates and examines attributes for project folders.

Syntax

```
project_folder [folderName] [-folderlist] [-filelist] [-printout] [-add] [-remove] [-r]
    [-tooltag] [-toolargs]
```

The following table describes the command options.

Option	Description
<i>folderName</i>	Specifies the name of the folder for which attributes are examined.
-folderlist	Lists folders contained in the specified project folder.
-filelist	Lists files contained in the specified project folder.
-printout	Prints the specified project folder hierarchy including its files.
-add	Adds a new project folder.
-remove	Removes the specified project folder.
-r	Removes the specified project folder and all its containing sub-folders. Files are removed from the project folder, but are not deleted.

Examples

Add a folder and list the files added to a project folder.

```
% project_folder -add newfolder
```

```
% project_folder -filelist newfolder
```

recording

Allows you to record and store the Tcl commands generated when you work on your projects in the GUI. You can use this command for creating job scripts. The complete syntax for the recording command is:

```
recording  
-on|-off  
-file [historyLogFile]  
-save [historyLogFile]  
-state  
-edit [filename]
```

In the command line:

- **on|off** – turns Tcl command recording on or off. Recording mode is off by default.
- **file** – if you specify a history log file name, this option uses the specified file in which to store the recorded Tcl commands for the current session. If you do not specify a history log name, reports the name of the current history log file.
- **save** – if you do not specify a file name, updates the current history log. If you specify a history log file name, saves Tcl command history to the specified file.
- **state** – returns the Boolean value of recording mode.
- **-edit** - displays the Tcl command log file in a text editor.

Examples

Turn on recording mode and save the Tcl commands in the `cpu_tcl_log` file created.

```
% recording -on  
% recording -file cpu_tcl_log
```

report_clocks

Reports the clocks in the design database.

Syntax

```
report_clocks -netlist [srsNetlistFile] [-csv_format] [-out fileName]
```

Arguments and Options

srsNetlistFile

The name of the srs netlist file. If this optional argument is not specified, the netlist file is taken from the active project implementation.

-csv_format

Displays the report in spread-sheet format.

-out

Specifies the name of the output report file (default name is *design-Name_clk.rpt*).

Description

The `report_clocks` command generates a report of the clocks found in the design database. The report includes a listing of the clock domain, parent clock, and clock type for each clock. If the `-csv_format` option is included, the report is output in spread-sheet format.

Examples

```
report_clocks c:/designs/mem_ctrl/mem_ctrl.srs -csv_format
```

run_tcl

The `run_tcl` command lets you synthesize your project using a Tcl script file from the Tcl Script window of the synthesis tool.

Syntax

```
run_tcl [ -fg ] tclFile
```

You can also use the following command:

```
source tclFile
```

These commands are equivalent.

The following table describes the `run_tcl` command options.

Option	Description
-fg	Synthesizes the project in foreground mode.
<i>TclFile</i>	Specifies the name of the Tcl file used to synthesize the project. To create a Tcl Script file, see Creating a Tcl Synthesis Script, on page 524 .

sd2fdc

Translates legacy FPGA timing constraints to Synopsys FPGA timing

Syntax

```
sd2fdc
```

Run it from the Tcl window in the synthesis tool.

See also

- [Converting SDC to FDC, on page 158](#) in the User Guide
- [sd2fdc Conversion, on page 143](#) in the *Reference Manual*

Examples of sdc2fdc Translation

The following are examples of feedback after running the command. For information about the translated FDC file and handling the error messages, see [sdc2fdc Conversion, on page 143](#) in the *Reference Manual*.

```
% sdc2fdc

INFO: Translation successful.
See:"D:/bugs/timing_88/clk_prior/scratch/FDC_constraints/rev_2
/top_translated.fdc"
Replace your current *.sdc files with this one.

INFO: Automatically updating your project to reflect the new
constraint file(s)
Do "Ctrl+S" to save the new settings.

% sdc2fdc

ERROR: Bad -from list for define_false_path: {my_inst}
Missing qualifier(s) (i: p: n: ...)
ERROR: Translation problems were found.
See:"D:/bugs/timing_88/clk_prior/scratch/FDC_constraints/rev_2
/top_translate.log" for details.
_translate.log

ERROR: Bad -from list for define_false_path {my_inst}
Missing qualifier(s) (i: p: n: ...)

"define_false_path -from {my_inst} -to i:abc.def.g_reg
-through {n:bar}"
Synplicity SDC source file: D:/bugs/timing_88/clk_prior/scratch
/top.sdc.
Line number: 79
```

set_option

The `set_option` command sets options for the technology (device) as well as for the design project.

Syntax

```
set_option -optionName optionValue
```

For syntax and descriptions of the options and related values, see one of the following tables:

- [Device Options for set_option/get_option](#)
- [Project Options for set_option/get_option](#)

Device Options for set_option/get_option

The following table lists *generic* device arguments for the technology, part, and speed grade. These are the options on the Implementation Options-> Device tab.

Information on all other Implementation Options tabs are listed in the next section, [Project Options for set_option/get_option, on page 68](#).

Option Name	Description
-technology <i>parameter</i>	Sets the target technology for the implementation. <i>parameter</i> is the string for the vendor architecture. Check the Device panel in the GUI or see Device Panel, on page 215 , for a list of supported families.
-part <i>part_name</i>	Specifies a part for the implementation. Check the Device panel of the Implementation Options dialog box (see Device Panel, on page 215) for available choices.
-speed_grade <i>-value</i>	Sets the speed grade for the implementation. Check the Device panel of the Implementation Options dialog box (see Device Panel, on page 215) for available choices.
-package <i>value</i>	Sets the package for the implementation. This option is not available for certain vendor families, because it is set in the place-and-route software. Check the Device panel of the Implementation Options dialog box (see Device Panel, on page 215) for available choices.
-grade <i>-value</i>	Same as -speed_grade. Included for backwards compatibility.

In general, device options are technology-specific, or have technology-specific defaults or limitations. For vendor-specific details, see *synhooks* [File Syntax, on page 370](#).

Project Options for set_option/get_option

Below is a list of options for the set_option and get_option commands. Click on the option below for the corresponding description and GUI equivalents. Options set through the Device tab are listed in [Device Options for set_option/get_option](#), on page 67.

analysis_constraint	areadelay
area_delay_percent	auto_constrain_io
autosm	beta_vfeatures
block	compiler_compatible
compiler_constraint	constraint
default_enum_encoding	disable_io_insertion
dup	enable64bit
fanout_limit	frequency auto
frequency	globalthreshold
hdl_param	hdl_define
help	identify_debug_mode
ignore_undefined_libs	include_path
job (PR)	libext
library_path	log_file
looplimit	maxfan
maxfan_hard	max_parallel_jobs
multi_file_compilation_unit	num_critical_paths
num_startend_points	opcond
project_relative_includes	preserve_registers
report_path	reporting_reportType
resolve_multiple_driver	resource_sharing
result_file	retiming
run_prop_extract	RWCheckOnRam

safe_case	symbolic_fsm_compiler
synthesis_onoff_pragma	top_module
update_models_cp	use_fsm_explorer
vlog_std	write_apr_constraint
write_verilog	write_vhdl

Option	Description	GUI Equivalent
-analysis_constraint <i>path/filename.adc</i>	Specifies the analysis design constraint file (adc) you can use to modify constraints for the stand-alone Timing Analyst only.	Constraint File section on the Timing Report Generation Parameters dialog box
-areadelay <i>percentValue</i>	Sets the percentage of paths you want optimized. This option is available only in certain device technologies.	Percent of design to optimize for timing, Device Panel
-area_delay_percent <i>percentValue</i>		
-auto_constrain_io 1 0	<p>Determines whether default constraints are used for I/O ports that do not have user-defined constraints.</p> <p>When disabled, only <code>define_input_delay</code> or <code>define_output_delay</code> constraints are considered during synthesis or forward-annotated after synthesis.</p> <p>When enabled, the software considers any explicit <code>define_input_delay</code> or <code>define_output_delay</code> constraints, as before.</p>	Use clock period for unconstrained IO check box, Constraints Panel
-autosm 1 0	Enables/disables the FSM compiler.	FSM Compiler check box, Options Panel
-symbolic_fsm_compiler 1 0		

Option	Description	GUI Equivalent
-beta_vfeatures 1 0	Enables/disables the use of Verilog compiler beta features.	Beta Features for Verilog, Compiler Directives and Design Parameters
-block 1 0 -disable_io_insertion 1 0	Enables/disables I/O insertion in some technologies.	Disable I/O Insertion check box, Device Panel
-compiler_compatible 1 0	Disables pushing of tristates across process/block boundaries.	<i>Complement of the Push Tristates Across Process/Block Boundaries</i> check box, VHDL Panel and Compiler Directives and Design Parameters
-compiler_constraint constraintFile	When multiple constraint files are defined, specify which constraint files are to be used from the Constraints tab of the Implementation Options panel.	Constraints Files, Constraints Panel
-constraint -option	Manipulates constraint files in the project: -enable/disable <i>filename</i> – adds or removes constraint file from active implementation -list – lists all enabled constraint files in active implementation -all – enables all constraint files in active implementation -clear – disables all constraint files in active implementation	Constraint Files, Constraints Panel

Option	Description	GUI Equivalent
-default_enum_encoding default onehot gray sequential	(VHDL only) Sets the default for enumerated types.	Default Enum Encoding, VHDL panel (see VHDL Panel and Compiler Directives and Design Parameters)
-disable_io_insertion 1 0 -block 1 0	Enables/disables I/O insertion in some technologies. For more information about the impact of using this command, see syn_insert_pad , on page 96.	Disable I/O Insertion, Device Panel
-dup	For Verilog designs, allows the use of duplicate module names. When true, the last definition of the module is used by the software and any previous definitions are ignored. You should not use duplicate module names in your Verilog design, therefore, this option is disabled by default. However, if you need to, you can allow for duplicate modules by setting this option to 1.	Allow Duplicate Modules, Compiler Directives and Design Parameters
-enable64bit 1 0	Enables/disables the 64-bit mapping switch. When enabled, this switch allows you to run client programs in 64-bit mode, if available on your system.	Enable 64-bit Synthesis, Options Panel
-fanout_limit <i>value</i> -maxfan <i>value</i>	Sets the fanout limit guideline for the current project.	Fanout Guide, Device Panel
-frequency <i>value</i>	Sets the global frequency.	Frequency, Constraints Panel
-frequency auto	Enables/disables auto constraints.	Auto Constrain, Constraints Panel

Option	Description	GUI Equivalent
-globalthreshold <i>value</i>	<p>This option applies only to the following Microsemi technologies:</p> <ul style="list-style-type: none"> • FUSION • IGLOO/IGLOOE/IGLOO+ • ProASIC3/3E/3L <p>Sets the minimum number of fanout loads. Signals that exceed the load value are promoted to global signals. Global buffers are assigned by the synthesis tool to drive the global signals.</p>	Device Panel
-hdl_define	For Verilog designs; used for extracting design parameters and entering compiler directives.	Compiler Directives and Design Parameters, Compiler Directives and Design Parameters
-hdl_param	Shows or sets HDL parameter overrides. See hdl_param , on page 44 for command syntax.	Use this command in the Tcl window of the UI.
-help	<p>This option is useful for getting syntax help on the various implementation options used for compiling and mapping a design. For examples, see help for set_option, on page 83.</p>	Use this command in the Tcl window of the UI.
-identify_debug_mode 1 0	When set option to 1, creates an Identify implementation in the Project view. Then, you can launch the Identify Instrumentor or Debugger from within the FPGA synthesis tools.	<p>Select the Identify implementation, then launch:</p> <ul style="list-style-type: none"> • Launch Identify Instrumentor <p>or</p> <ul style="list-style-type: none"> • Launch Identify Debugger

Option	Description	GUI Equivalent
-ignore_undefined_libs 1 0	(VHDL only) When enabled (default), the compiler will ignore any declared library files not included with the source file. In previous releases, the missing library file would cause the synthesis tool to error out. To set this option to error out when a library file is missing (as in previous releases), use 0 for the command value.	Not available in the UI
-include_path <i>path</i> ./extra_input/	(Verilog only) Defines the search path used by the 'include commands in Verilog design files. Argument <i>path</i> is a string that is a semicolon-delimited list of directories where the included design files can be found. The software searches for include files in the following order: <ul style="list-style-type: none"> • First, the source file directory. • Then, looks in the included path directory order and stops at the first occurrence of the included file it finds. • Finally, the project directory. The include paths are relative. Use the <code>project_relative_includes</code> option to update older project files. The archive utility allows you to add the <code>extra_input</code> directory path for all include files and copies them to your project. Use the <code>Add extra input path to project</code> option on the Un-Archive Utility dialog box.	Include Path Order, Verilog panel (see Compiler Directives and Design Parameters, on page 231)
-job <i>PR_job_name</i> -option enable_run 1 0	If enabled, runs the specified place-and-route job with the appropriate vendor-specific place-and-route tool after synthesis.	Specify the place-and-route job you want to run for the specified implementation. See Place and Route Panel .

Option	Description	GUI Equivalent
-libext <i>.libextName1 .libextName2 ...</i>	<p>Adds library extensions to Verilog library files included in your design for the project and searches the directory paths you specified that contain these Verilog library files. To use library extensions, see Using Library Extensions for Verilog Library Files, on page 40 in the <i>User Guide</i>.</p>	<p>Library Extensions (space separated) for each unique file extension, Compiler Directives and Design Parameters.</p>
-library_path <i>directory_pathname</i>	<p>For Verilog designs, specifies the paths to the directories which contain the library files to be included in your design for the project. Defines the search path used by the tool to include all the Verilog design files for your project. The argument <i>directory_pathname</i> is a string that specifies the directories where these included library files can be found. The software searches for all included Verilog files and the tool determines the top-level module.</p>	<p>Library Directories on Compiler Directives and Design Parameters.</p>
-log_file <i>logFileName</i>	<p>Allows you to change the name for a default log file (both the srr and htm files). For example:</p> <pre>set_option -log_file test</pre> <p>generates the following files in the Implementation Directory after synthesis is run:</p> <ul style="list-style-type: none"> • test.htm • synlog\test_premap.srr • synlog\test_fpga_mapper.srr • synlog\test_fpga_mapper.srr_Min 	<p>Enter command from the Tcl window</p>

Option	Description	GUI Equivalent
looplimit <i>loopLimitValue</i>	Allows you to override the default compiler loop limit value of 2000 in the RTL. You can also apply loop limits using the Verilog <code>loop_limit</code> or the VHDL <code>syn_looplimit</code> directive. For details about these directives, see loop_limit, on page 37 and syn_looplimit, on page 113 in the <i>Attribute Reference</i> .	Loop Limit, Compiler Directives and Design Parameters and VHDL Panel .
-maxfan <i>value</i> -fanout_guide <i>value</i>	Sets the fanout limit for the current project. The limit is value is guideline for the tool rather than a hard limit.	Fanout Guide, Device Panel
-maxfan_hard 1	This option specifies that the -maxfan value is a hard fanout limit that the synthesis tool must not exceed it.	Hard Limit to Fanout, Device Panel
max_parallel_jobs <i>n</i>	Lets you run multiprocessing with compile points. This allows the synthesis software to run multiple, independent compile point jobs simultaneously, providing additional runtime improvements for the compile point synthesis flow. For information on setting the maximum number of parallel synthesis jobs, see Setting Number of Parallel Jobs, on page 523 in the <i>User Guide</i> .	Maximum number of parallel mapper jobs, on the Configure Compile Point Process dialog box.
-multi_file_compilation_unit 1 0	When you enable the Multiple File Compilation Unit switch, the Verilog compiler uses the compilation unit for modules defined in multiple files.	Compiler Directives and Design Parameters
-num_critical_paths <i>value</i>	Specifies the number of critical paths to report in the timing report.	Number of Critical Paths, Timing Report Panel

Option		Description	GUI Equivalent
-num_startend_points	<i>value</i>	Specifies the number of start and end points to include when reporting paths with the worst slack in the timing report.	Number of Start/End Points, Timing Report Panel . Number of Start/End Points, Timing Report Generation dialog box.
-opcond	<i>value</i>	This option applies only to the Microsemi Fusion and IGLOO families of technologies. Sets the operating condition for device performance in the areas of optimization, timing analysis, and timing reports. Values are Default, MIL-WC, IND-WC, COM-WC, and Automotive-WC. See Operating Condition Device Option, on page 415 for more information.	Device Panel
-preserve_registers	1 0	When enabled, the software uses less restrictive register optimizations during synthesis if area is not as great a concern for your device. The default for this option is disabled (0).	Conservative Register Optimization switch on the Device Panel
-project_relative_includes	1 0	Enables/disables the Verilog include statement to be relative to the project, rather than a verilog file. For projects built with software after 8.0, the include statement is no longer relative to the files but is relative to the project: project_relative (1). See Updating Verilog Include Paths in Older Project Files, on page 63 in the <i>User Guide</i> for information about updating older project files.	Include Path Order, Compiler Directives and Design Parameters
-report_path	<i>integer</i>	Available for Microsemi Fusion, and IGLOO family of technologies. Sets the maximum number of critical paths in a forward-annotated SDF constraint file	Max Number of Critical Paths in SDF, Device Panel

Option	Description	GUI Equivalent
-reporting_reportType	Sets parameters for the stand-alone Timing Analyst report. See Timing Report Parameters for set_option , on page 80 for details.	Analysis->Timing Analyst command: Timing Report Generation Parameters
-resolve_multiple_driver 1 0	When a net is driven by a VCC or GND and active drivers, enable this option to connect the net to the VCC or GND driver. The default for this option is disabled (0). See Resolve Mixed Drivers Option , on page 84 for details.	Resolve Multiple Drivers, Device Panel
-resource_sharing 1 0	Enables or disables resource sharing globally. This is a compiler-specific optimization, and does not affect resource sharing in the mapper. To enable or disable individual modules, use the syn_sharing directive.	Resource Sharing, Device Panel
-result_file filename	Specifies the name of the results file.	Result File Name and Result Format, Implementation Results Panel
-retiming 1 0	When enabled (1), registers may be moved into combinational logic to improve performance. The default value is 0 (disabled).	Retiming, Device Panel
-run_prop_extract 1 0	Enables/disables the annotation of certain generated properties relating to clocks and expansion onto the RTL view. This enables the Tcl expand and find commands to work correctly with clock properties.	Options Panel

Option	Description	GUI Equivalent
-rw_check_on_ram 1 0	<p>Enabling this option automatically inserts bypass logic when required to prevent simulation mismatch in read-during-write scenarios. For asynchronous clocks, the tool will not generate bypass logic which can cause unintended CDC paths between the clocks.</p> <p>For more information about using this option in conjunction with the <code>syn_ramstyle</code> attribute, see syn_ramstyle, on page 173.</p>	Automatic Read/Write Check Insertion for RAM, Device Panel
-safe_case 1 0	<p>When enabled, the high reliability safe case option turns off sequential optimizations for counters, FSM, and sequential logic to increase the circuit's reliability.</p>	Preserve and Decode Unreachable States (FSM, Counters, Sequential Logic), High Reliability Panel

Option	Description	GUI Equivalent
-symbolic_fsm_compiler 1 0 -autosm 1 0	<p>Enables/disables the FSM compiler. Controls the use of FSM synthesis for state machines. The default is false (FSM Compiler disabled). Value can be 1 or true, 0 or false.</p> <p>When this option is true, the FSM Compiler automatically recognizes and optimizes state machines in the design. The FSM Compiler extracts the state machines as symbolic graphs, and then optimizes them by re-encoding the state representations and generating a better logic optimization starting point for the state machines.</p> <p>However, if you turn off sequential optimizations for the design, FSM Compiler and/or the <code>syn_state_machine</code> directive and <code>syn_encoding</code> attribute are effectively disabled.</p> <p>See -no_sequential_opt 1 0 for more information on turning off sequential optimizations.</p>	<p>FSM Compiler check box, Device Panel</p>
-synthesis_onoff_pragma 1 0	<p>Determines whether code between synthesis on/off directives is ignored.</p> <p>When enabled, the software ignores any VHDL code between <code>synthesis_on</code> and <code>synthesis_off</code> directives. It treats these third-party directives like <code>translate_on/off</code> directives (see translate_off/translate_on, on page 229 for details).</p>	<p>Synthesis on/off Implemented as Translate on/Off, VHDL Panel</p>

Option	Description	GUI Equivalent
-top_module <i>name</i>	Specifies the top-level module. If the top-level entity does not use the default work library to compile the VHDL files, you must specify the library file where the top-level entity can be found. To do this, the top-level entity name must be preceded by the VHDL library followed by the dot (.).	Top-level Entity/Module, VHDL Panel or Compiler Directives and Design Parameters
-update_models_cp 1 0	Determines whether (1) or not (0) changes inside a compile point can cause the compile point (or top-level) containing it to change accordingly.	Update Compile Point Timing Data, Device Panel
-use_fsm_explorer 1 0	Enables/disables the FSM Explorer.	FSM Explorer, Device Panel
-vlog_std v2001 v95 sysv	The default Verilog standard for new projects is SystemVerilog. Turning off both options in the Verilog panel defaults to v95.	Verilog 2001, SystemVerilog, Compiler Directives and Design Parameters
-write_apr_constraint 1 0	Writes vendor-specific constraint files.	Write Vendor Constraint File, Implementation Results Panel
-write_verilog 1 0	Writes Verilog or VHDL mapped netlists.	Write Mapped Verilog/VHDL Netlist, Implementation Results Panel
-write_vhdl 1 0		

Timing Report Parameters for set_option

The following lists the parameters for the stand-alone timing report (ta file).

<code>async_clock</code>	<code>margin</code>
<code>ctd</code>	<code>netlist</code>
<code>filename</code>	<code>number_paths</code>
<code>filter</code>	<code>output_srm</code>
<code>gen_output_srm</code>	

Reporting Option	Description
-reporting_async_clock	Generates a report for paths that cross between clock groups using the stand-alone Timing Analyst.
-reporting_ctd slack end_point off	<p>Controls how the <i>design_ctd.txt</i> (correlation timing dump) file is generated when the Timing Analyst is run. You can specify one of the following values:</p> <ul style="list-style-type: none"> • slack – The timing information in the <i>ctd</i> file is sorted by slack. This is the default. • end_point – The timing information in the <i>ctd</i> file is sorted by end points. • off – Turns off generating the <i>ctd</i> file. <p>The <i>ctd</i> file contains a timing summary of the design that is used by the Timing Report View to display and analyze the synthesis timing for the design and correlate this synthesis timing with the P&R timing in the GUI.</p>
-reporting_filename <i>filename.ta</i>	Specifies the standard timing report file (<i>ta</i>) generated from the stand-alone Timing Analyst.

Reporting Option	Description
-reporting_filter <i>filter options</i>	<p>Generates the standard timing report based on the filter options you specify for paths, such as:</p> <ul style="list-style-type: none"> • From points • Through points • To points <p>For more information, see:</p> <ul style="list-style-type: none"> • Timing Report Generation Parameters, on page 292. • Combining Path Filters for the Timing Analyzer, on page 296 • Timing Analyzer Through Points, on page 295. • Specifying From, To, and Through Points, on page 187.
-reporting_gen_output_srm 1 0	Specifies the new name of the output SRM File when you change the default name. If this option is set to 1, this new name is used for the output srm file after you run the stand-alone Timing Analyst.
-reporting_margin <i>slackValue</i>	You can specify a slack margin to obtain a range of paths within the worst slack time for the design after you run the stand-alone Timing Analyst.
-reporting_netlist <i>filename.srm</i>	Specifies the associated gate-level netlist file (srm) generated from the stand-alone Timing Analyst.
-reporting_number_path <i>numberOfPaths</i>	You can specify the number of critical paths to report after you run the stand-alone Timing Analyst.
-reporting_output_srm 1 0	Allows you to change the name of the output srm file. If you enable the output SRM File option, you can change this default name.

For GUI equivalent switches for these parameters, see [Timing Report Generation Parameters, on page 292.](#)

help for set_option

This option is useful for getting syntax help on the various implementation options used for compiling and mapping a design, especially since this list of options keeps growing.

Syntax

```
% set_option -help
```

Usage:

```
set_option optionName optionValue [-help [value]]
```

Where:

- *optionName*—specifies the option name.
- *optionValue*—specifies the option value.
- -help [*value*]—to get help on options. Use:
 - -help * for the list of options
 - -help *optionName* for a description of the option

Examples

To list all option commands in the Tcl window:

```
set_option -help *
```

To list all option commands beginning with the letters sy in the Tcl window:

```
% set_option -help sy*  
  
symbolic_fsm_compiler  
synthesis_onoff_pragma
```

To get help on a specific option in the Tcl window:

```
% set_option -help symbolic_fsm_compiler  
  
Extracts and optimizes finite state machines
```

Use the following Tcl commands to print a description of the options:

```
% set_option -help c*
% set hl [set_option -help c*]
% puts $hl
% foreach option $hl { puts "$option:\t [set_option -help
$option]"; }
```

This example will print a list of `set_option` options that begin with the letter c.

Resolve Mixed Drivers Option

Use the Resolve Mixed Drivers option when mapping errors are generated for input nets with mixed drivers. You might encounter the following messages in the log file:

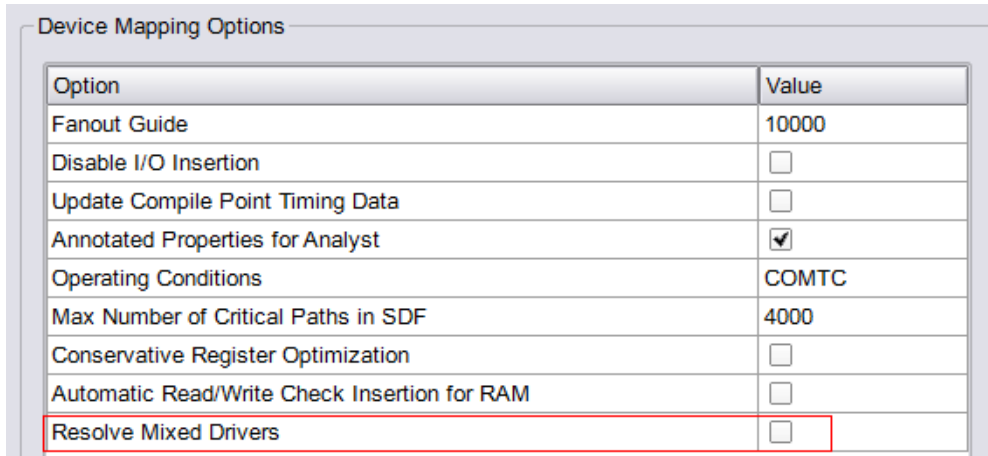
```
@A:BN313 | Found mixed driver on pin pin:data_out inst:dpram_lut3
of work.dpram(verilog), use option "Resolve Mixed Drivers" in
"Device" tab of "Implementation Options" to automatically resolve
this
@E:BN314 | Net "GND" in work.test(verilog) has mixed drivers

@A:BN313 | Found mixed driver on pin pin:Q[0] inst:dffl.q of
PrimLib.sdffr(prim), use option "Resolve Mixed Drivers" in
"Device" tab of "Implementation Options" to automatically resolve
this
@E:BN314) | Net "VCC" in work.test(rtl) has mixed drivers
```

Whenever a constant net (GND or VCC) and an active net are driving the same output net, enable the Resolve Mixed Drivers option so that synthesis can proceed. To set this switch:

- Check Resolve Mixed Drivers on the Device tab of the Implementation Options panel.

- Use the Tcl command, `set_option -resolve_multiple_driver 1`.



By default this option is disabled and set to:

```
set_option -resolve_multiple_driver 0.
```

When you rerun synthesis, you should now see messages like the following in the log file:

```
@W:BN312 | Resolving mixed driver on net GND, connecting output
pin:data_out inst:dpram_lut3 of work.dpram(verilog) to GND
@N:BN116 | Removing sequential instance dpram_lut3.dout of
view:PrimLib.dffe(prim) because there are no references to its
outputs
@N:BN116 | Removing sequential instance dpram_lut3.mem of
view:PrimLib.raml(prim) because there are no references to its
outputs

@W:BN312 | Resolving mixed driver on net VCC, connecting output
pin:Q[0] inst:dff1.q of PrimLib.sdffr(prim) to VCC
@N:BN116 | Removing sequential instance dff1.q of
view:PrimLib.sdffr(prim) because there are no references to its
outputs
```

Example – Active Net and Constant GND Driving Output Net (Verilog)

```

module test(clk,data_in,data_out,radd,wradd,wr,rd);
input clk,wr,rd;
input data_in;
input [5:0]radd,wradd;
output data_out;
// component instantiation for shift register module
shr1 srl_lut0 (
    .clk(clk),
    .sren(wr),
    .srin(data_in),
    .srout(data_out)
);
// Instantiation for ram
dpram dpram_lut3 (
    .clk(clk),
    .data_in(data_in),
    .data_out(data_out),
    .radd(radd),
    .wradd(wradd),
    .wr(wr),
    .rd(rd)
);

endmodule

module shr1 (clk,sren,srin,srout);
input clk;
input sren;
input srin;
output srout;

parameter width = 32;
reg [width-1:0] sr;

always@(posedge clk)
begin
    if (sren == 1)
    begin
        sr <= {sr[width-2:0], srin};
    end
end
// Constant net driving

// the output net
assign srout = 1'b0;

```

```
endmodule

module dpram(clk,data_in,data_out,radd,wradd,wr,rd);
input clk,wr,rd;
input data_in;
input [5:0]radd,wradd;
output data_out;

reg dout;
reg [0:0]mem[63 :0];

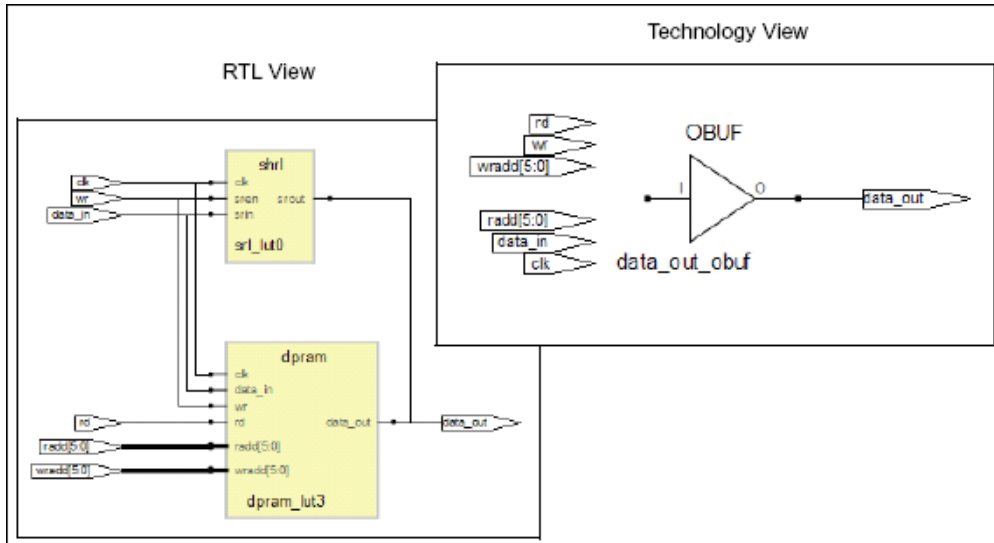
always @ (posedge clk)
begin
    if(wr)
        mem[wradd] <= data_in;
end

always @ (posedge clk)
begin
    if(rd)
        dout <= mem[radd];
    end

assign data_out = dout;

endmodule
```

See the following RTL and Technology views; the Technology view shows the constant net tied to the output.



Example – Active Net and Constant VCC Driving Output Net (VHDL)

```

library ieee;
use ieee.std_logic_1164.all;

entity test is
port (clk,rst : in std_logic;
      sr_en : in std_logic;
      data : in std_logic;
      data_op : out std_logic );
end entity test;

architecture rtl of test is
component shr1
generic (sr_length : natural);
port (clk : in std_logic;
      sr_en : in std_logic;
      sr_ip : in std_logic;
      sr_op : out std_logic );
end component shr1;

component d_ff
port (data, clk, rst : in std_logic;
      q : out std_logic );
end component d_ff;

```

```

begin
-- instantiation of shift register
shift_register : shr1
generic map (sr_length => 64)
port map (clk => clk,
          sr_en => sr_en,
          sr_ip => data,
          sr_op => data_op );
-- instantiation of flipflop
dff1 : d_ff
port map (data => data,
          clk => clk,
          rst => rst,
          q => data_op );
end rtl;

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

entity shr1 is
generic (sr_length : natural);
port (clk : in std_logic;
      sr_en : in std_logic;
      sr_ip : in std_logic;
      sr_op : out std_logic );
end entity shr1;

architecture rtl of shr1 is
signal sr_reg : std_logic_vector(sr_length-1 downto 0);
begin
  shreg_lut: process (clk)
  begin
    if rising_edge(clk) then
      if sr_en = '1' then
        sr_reg <= sr_reg(sr_length-2 downto 0) & sr_ip;
      end if;
    end if;
  end process shreg_lut;
  -- Constant net driving output net
  sr_op <= '1';
end architecture rtl;

library IEEE;
use IEEE.std_logic_1164.all;

```

```

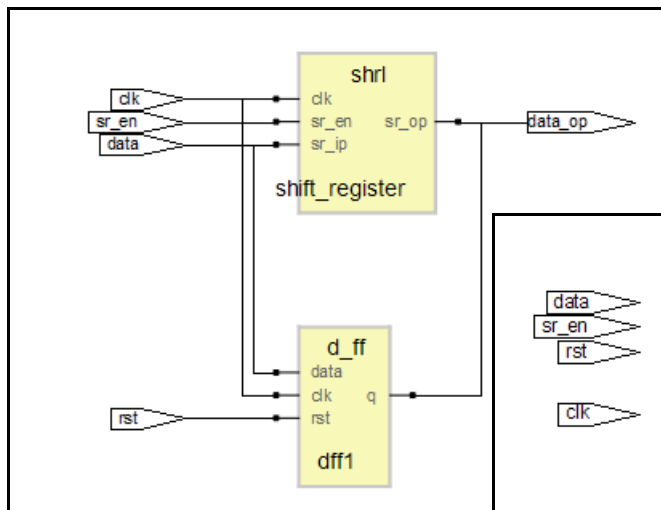
entity d_ff is
port (data, clk, rst : in std_logic;
      q : out std_logic );
end d_ff;

architecture behav of d_ff is
begin
  FF1:process (clk) begin
    if (clk'event and clk = '1') then
      if (rst = '1') then
        q <= '0';
      else q <= data;
      end if;
    end if;
  end process FF1;
end behav;

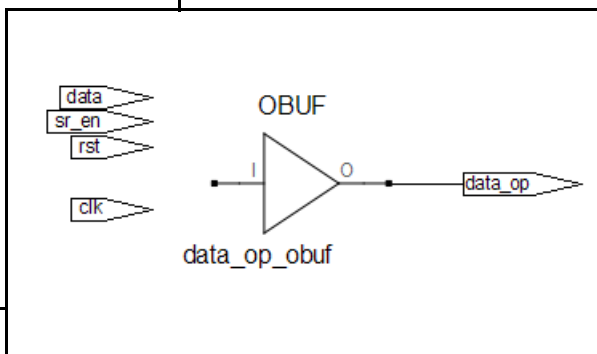
```

See the following RTL and Technology views; the Technology view shows the constant net tied to the output.

RTL View



Technology View



status_report

Writes out the results of reports displayed in the Project Status view after synthesizing a design.

Syntax

```
status_report -name reportName [-parameter reportSectionName]
[-csv] [-output_file fileName] [-msgtype msgStatus] [-status] [-help]
```

Examples

```
status_report -name area_report

status_report -name timing_report -csv -output_file reports

status_report -name area_report -parameter io_port

status_report -name run_status -msgtype warnings

status_report -name timing_report -help
```

Option	Description
-name <i>reportName</i>	The type of report to access. Use any of the following keywords for <i>reportName</i> : <ul style="list-style-type: none"> • area_report • timing_report • opt_report • cp_report • hier_area_report • run_status
-parameter <i>reportSectionName</i>	Specifies a specific section of the area, timing, or message reports to access. See Parameters, on page 92 for details of the appropriate keywords to use for the section names.
-csv	Generates the report as a comma-separated list.
-output_file <i>fileName</i>	Specifies the name of the file that writes out the report. If you do not specify an output file, the report is displayed in the Tcl window.

Option	Description
-msgtype <i>msgStatus</i>	Generates the number of messages found for the following types of messages: <ul style="list-style-type: none"> • Errors • Warnings • Notes
-status	Generates the status for a job. The results of the job can be specified with one of the following conditions: <ul style="list-style-type: none"> • Completed • Failed
-help	Allows you to get help on a parameter list. Use -help * for a list of parameters.

Parameters

For the area, timing, and message reports you can report results for specific sections by specifying the appropriate keywords for the -parameter argument.

Area Report section keywords for -parameter	io_port non_io_reg total_io_reg v_ram dsp_used total_luts
Timing Report section keywords for -parameter	clock_name req_freq est_freq slack
Run Status section keywords for -parameter	compiler premap fpga_mapper

For example:

```
% status_report -name area_report
I/O ports(io_port) 26
Non I/O Register bits(non_io_reg)242 (0%)
I/O Register bits(total_io_reg) 24
Block Rams(v_ram) 0 (1030)
DSP48s(dsp_used) 1 (2800)
LUTs(total_luts) 310 (0%)
```

Additional Reporting Commands

There are other commands available from the command line to report commonly-required information: `report_timing_summary`, `report_area`, and `report_opt`.

- Report Timing Summary

```
% report_timing_summary
Timing Summary
Clock Name Req Freq Est Freq Slack
eight_bit_uc|clock 198.9 MHz 169.1 MHz-0.887
```

- Report Area

```
% report_area
LUTs for combinational functions0
Non I/O Registers 0
I/O Pins 66
I/O registers 0
DSP Blocks 0 (256)
Memory Bits 32768
```

- Report Optimizations

```
% report_opt
Combined Clock Conversion 1 / 0
```

Messages Reporting Commands

Here are examples of commands available from the command line to report message information using the option: `run_status`.

```
%status_report -name run_status
{compiler {notes "8"}{warnings "0"}{errors "0"}}
  {job_status "Completed"}}
{fpga_mapper {notes "46"}{warnings "1"}{errors "0"}}
  {job_status "Completed"}}
{premap {notes "3"}{warnings "2"}{errors "0"}}
  {job_status "Completed"}}

% status_report - name run_status -msgtype warnings
{compiler {warnings "0"}}}
{fpga_mapper {warnings "10"}}}
{premap {warnings "0"}}}

% status_report - name run_status -parameter {compiler premap}
  -msgtype warnings
{compiler {warnings "0"}}}
{premap {warnings "0"}}}

% status_report -name run_status -parameter compiler -status
{compiler {job_status "Completed"}}}

% status_report -name run_status -parameter premap -status
```

syn_connect

Specifies the connectivity between the module/instance being monitored for the error with the error monitoring IP port or top-level port for the error monitoring module. It is used with the `syn_create_err_net` command and the `syn_radhardlevel` attribute to specify I/O redundancy in high-reliability designs.

Syntax

```
syn_connect
  -from {n:netName}
  -to {n:existingNetpath | t:inputPinEMIPpath | p:topErrorport}
```

-from {i:netName}	Specifies the new net name for the error flag connection used with the <code>syn_create_err_net</code> command. See syn_create_err_net , on page 95.
-to {n:existingNetpath t:inputPinEMIPpath p:topErrorport}	Specifies the path to an existing net, the input pin for the Error Monitoring IP (EMIP), or the top-level error port.

Example

```
syn_connect {-from {n:dwc_err0_net} -to {n:emp[0]}}
```

For Microsemi examples, see [RAM Inferencing for RTG4](#), on page 708.

Limitation

Currently, when using the `-to` argument with the `t: | p:` option, an existing net must be present that feeds into the ports.

syn_create_err_net

Specifies the connectivity between the module/instance being monitored for the error with the error monitoring IP port or top-level port for the error monitoring module. It is used with the `syn_connect` command and the `syn_rad-hardlevel` attribute to specify I/O redundancy in high-reliability designs.

This is the syntax:

```

sync_create_err_net -name {netName}
  -inst {i:highRelInstance}
  -err_pipe_num {numPipelineStages}
  -err_clk {n:clkSourceForPipelineStages}
  -err_reset {n:resetSourceForPipelineStages}
  -err_set {n:setSourceForPipelineStages}
  -err_enable {n:enableSourceForPipelineStages}
  -err_synch {synchValue}

```

<i>netName</i>	Specifies the new net name for the error flag connection.
-inst <i>highRelInstantiation</i>	Specifies the high reliability instance being monitored for the error.
-err_pipe_num <i>{numPipelineStages}</i>	Specifies the number of pipeline stages using the retiming property. The default value is 0. Note: If <code>err_pipe_num=N</code> , then the property <code>syn_allow_retiming=0</code> is set on the Nth stage of the pipeline register and <code>syn_allow_retiming=1</code> is set on the first (N-1)th stage of the pipeline registers by the synthesis tool. Where $N > 0$ and <code>err_clk</code> is specified.
-err_clk <i>{n:clkSourceforPipelineStages}</i>	Specifies the hierarchical path to the input pin for the clock. Note: You must define the <code>-err_clk</code> for the pipeline stages if you specify <code>-err_pipe_num</code> . Otherwise, the pipeline stages to stabilize the error signals are not added.
-err_reset <i>{n:resetSourceforPipelineStages}</i>	Specifies the hierarchical path to the input pin for the reset signal.
-err_set <i>{n:setSourceforPipelineStages}</i>	Specifies the hierarchical path to the input pin for the set signal.
-err_enable <i>{n:enableSourceforPipelineStages}</i>	Specifies the hierarchical path to the input pin for the enable signal.
-err_synch <i>{synchValue}</i>	Specifies the boolean value for synchronous or asynchronous set/reset. The default is TRUE.

Example

```
syn_create_err_net {-name {dwc_err0_net} -inst {i:inst1}
  -err_pipe_num {2} -err_clk {n:inst1.clk}
  -err_reset {n:inst1.rst} -err_synch {TRUE}}
```

For Microsemi examples, see [RAM Inferencing for RTG4, on page 708](#).

synplify_pro

Starts the FPGA synthesis tool and runs synthesis from the command line. The command to start the synthesis tool from the command line includes a number of command line options.

Syntax

```
synplify_pro
  [options ... ]
  [projectFile]
```

projectFile Specifies the project (prj) file to use. If no file is specified, the tool defaults to the last project file opened.

options Any of the command line options described in the next table. These options control tool action on startup and, in many cases, can be combined on the same command line. See the next table for a description of the *options* you can specify.

The following table describes the *options* you can specify:

Option	Description
-batch	<i>Synplify Pro (except node-locked)</i> Starts the synthesis tool in batch mode from the specified project or Tcl file without opening the Project window.
-compile	Compiles the project, but does not map it.
-evalhostid	Reports host ID for node-locked and floating licenses.
-help	Lists available command line options and descriptions.
-history filename	Records all Tcl commands and writes them to the specified history log file when the command exits.

Option	Description
-identify_dir <i>dir</i>	Specifies the location of the Identify installation directory for launching the Identify tool set. The installation path specified appears in the Configure Identify Launch dialog box (Options->Configure Identify Launch).
-impl <i>impName</i>	Runs only the specified implementation. You can use this option in conjunction with the <code>-batch</code> keyword.
-ip_license_wait <i>waitTime</i>	<p>Specifies how long to wait for a Synopsys DesignWare IP license when one is not immediately available. If you do not specify the <code>-ip_license_wait</code> option, license queuing is not enabled.</p> <p>If all requested licenses are checked out or if the specified wait time elapses, the tool excludes the IP and continues to process the rest of the design. Any IP block without a license is treated either as an error or a black box.</p> <p>License queuing allows you to wait until a license becomes available or specify a wait time in seconds. You can use this option in conjunction with the <code>-batch</code> keyword. For details, see Queuing Licenses, on page 518 in the <i>User Guide</i>.</p> <p>The <i>waitTime</i> value determines license queuing and sets a maximum wait time:</p> <ul style="list-style-type: none"> • Undefined or 0 = Queuing off • 1 = Queuing enabled, indefinite wait time • >1 = Queuing enabled for the specified time
-license_release	<p>Releases FPGA synthesis licenses for a session after the place-and-route job is launched. The software allows place and route to continue running even after exiting the synthesis tool so that it does not consume an FPGA license.</p> <p>This command option must be run in batch mode. Specify the following command:</p> <pre><i>toolName</i> -batch -license_release</pre> <p>For details, see Releasing the Synthesis License During Place and Route, on page 550.</p>
-licensetype <i>featureName</i>	Specifies a license if you work in an environment with multiple Synopsys FPGA licenses. You can use this option in conjunction with the <code>-batch</code> keyword.

Option	Description
-license_wait <i>waitTime</i>	<p>Specifies how long to wait for a Synopsys FPGA license. If you do not specify the <code>-license_wait</code> option, license queuing is not enabled.</p> <p>License queuing allows you to wait until a license becomes available or specify a wait time in seconds. You can use this option in conjunction with the <code>-batch</code> keyword. For details, see Queuing Licenses, on page 518 in the <i>User Guide</i>.</p> <p>The <i>waitTime</i> value determines license queuing and sets a maximum wait time in seconds:</p> <ul style="list-style-type: none"> • Undefined or 0 = Queuing off • 1 = Queuing enabled, indefinite wait time • >1 = Queuing enabled for the specified wait time
-log <i>filename</i>	Writes all output to the specified log file.
-runall	Runs all the implementations in the project file (the Synplify tool supports only a single implementation).
-shell	<p>Starts synthesis tool in shell mode.</p> <p>Note: The FPGA synthesis tools only support the <code>-shell</code> option on UNIX and Linux platforms.</p>
-tcl <i>prjFile</i> <i>Tclscript</i>	Starts the synthesis tool in the graphical user interface using the specified project or Tcl file.
-tclcmd <i>command</i>	Specifies Tcl command to be executed on startup.
-verbose_log	Writes messages to stdout.log in verbose mode.
-version	Reports version of specified synthesis tool.

Tcl Command Categories

The following tables group Tcl commands together by type or functionality.

[Synthesis Commands, on page 100](#)

[Log File Commands, on page 100](#)

Synthesis Commands

add_file	add_folder
command_history	constraint_file
get_env	get_option
hdl_define	hdl_param
impl	job
open_design	open_file
partdata	project
project_data	project_file
project_folder	recording
run_tcl	set_option
status_report	

Log File Commands

These Tcl commands let you filter messages in the log file.

log_filter	Lets you filter errors, notes, and warning messages.
log_report	Lets you write out the results of the log_filter command to a file.

CHAPTER 3

Tcl Find, Expand, and Collection Commands

The FPGA synthesis software includes powerful search functionality in the Tcl find and expand commands. Objects located by these commands can be grouped into collections and manipulated. The following sections describe the commands and collections in detail:

- [find, on page 102](#)
- [find -filter, on page 116](#)
- [expand, on page 123](#)
- [Collection Commands, on page 126](#)
- [Query Commands, on page 136](#)
- [Synopsys Standard Collection Commands, on page 160](#)

find

The Tcl find command identifies design objects based on specified criteria. Use this command to locate multiple objects with a common characteristic. If you want to locate objects that share connectivity, use the expand command instead of the find command ([expand, on page 123](#)).

You can specify the find command from the SCOPE environment or enter it as a Tcl command. This command operates on the RTL database.

You can define objects identified by find as a group or *collection*, and operate on all the objects in the collection at the same time. To do this, you embed the find command as part of a collection creation or manipulation command to do this in a single step. The combination of find and collection commands provides you with very powerful functionality to operate on and manipulate multiple design objects simultaneously.

Version L-2016.09 of the software includes a beta version of find, as part of the beta version of HDL Analyst.

The table summarizes where to find detailed information:

For ...	See ...
Command syntax	Tcl Find Syntax, on page 103
Syntax details: object types, expressions, case sensitivity, and special characters	Tcl Find Command Object Types, on page 106 Regular Expressions, Wildcards, and Special Characters, on page 106 Tcl Find Command Case Sensitivity, on page 108
Examples of find syntax	Demos and Examples button, accessible from the tool UI Tcl Find Syntax Examples, on page 112
Filtering find searches by property	find -filter, on page 116 Find Filter Properties, on page 117 Refining Tcl Find Results with -filter, on page 140 in the <i>User Guide</i>.
Using find search patterns and using find in collections	Finding Objects with Tcl find and expand, on page 138 in the <i>User Guide</i>.

Tcl Find Syntax

```
find [-objectType] [pattern]
      [-in $collectionName | listName]
      [-hier] [-hsc character]
      [-regexp] [-nocase] [-exact]
      [-print]
      [-namespace techview | netlist]
      [-flat]
      [-leaf]
      [-rtl | -tech]
      [-filter expression]
      [-seq]
```

If used, the `-filter` option must be specified as the last argument in the command. Descriptions of each command option are listed alphabetically in the following table.

Argument	Description
-objectType <i>pattern</i>	Specifies the type of object to be found: view, inst, port, pin, net or seq. The object type must be preceded by the appropriate prefix, as described in Tcl Find Command Object Types, on page 106 . <i>pattern</i> specifies the search pattern to be matched, and can include the * and ? wildcard characters.
-exact	Disables simple pattern matching. Use it to search for objects that contain the * and ? wildcard characters. You cannot use this argument with <code>-nocase</code> or <code>-regexp</code> . See Regular Expressions, Wildcards, and Special Characters, on page 106 for additional information.
-filter <i>expression</i>	Refines the results of find further, by filtering the results by the specified object property. For details about the syntax, refer to find -filter, on page 116 . If you use the <code>-filter</code> option, it must be specified as the last argument to the find command.
-flat	Extends the search to all levels, but with <code>-flat</code> , the * wildcard character matches hierarchy separators as well as characters. This means that the following example finds instance a1_fft at the current level as well as the hierarchical instance a1.fft: <pre>find -seq -flat a1*fft</pre>

Argument	Description
-hier	<p>Searches for the pattern from every level of hierarchy, instead of just the top level. By default, the search occurs from the top-level hierarchy for the given pattern; this option allows you to search for the pattern from every level of hierarchy.</p> <p>When -hier is not specified, the search is limited to the current view and wildcards do not match the hierarchy delimiter character. You can still traverse downward through the hierarchy by adding the delimiter in the pattern. Thus an asterisk (*) matches any object at the current level, but *.* matches any object one level below the current view. For more about wildcard characters, see Regular Expressions, Wildcards, and Special Characters, on page 106.</p>
-hsc <i>character</i>	<p>Extends the search downward through each level of the local hierarchy, instead of limiting the search to the current view. The default hierarchy separator for the search is the period (.). To specify another hierarchy character, use the -hsc option. Use this option when the pattern is ambiguous. For example, with the default separator, block1.u1 could match either instance u1 in block1 or the object named block1.u1. Using a " " separator character eliminates the ambiguity. If you specify find -hier -hsc " " [block1 u1], the command finds hierarchical instance u1 in the block, while find -hier -hsc " " [block1.u1] finds the object.</p> <p>See Tcl Syntax Guidelines for Constraint Files, on page 50 for more information.</p>
-in \$collectionName/ <i>listName</i>	Restricts the search to the specified list or collection.
-leaf	Returns only non-hierarchical instances.
-nocase	<p>Ignores case when matching patterns. The default is to take case into account (-case). You cannot use the -nocase argument with -exact or -regexp. See Tcl Find Command Case Sensitivity, on page 108 for more information.</p>
-namespace techview netlist	<p>Determines the database to search for the find operation.</p> <ul style="list-style-type: none"> techview searches the mapped (srm) database. This is the default. netlist searches the output netlist. <p>This option is not available for an RTL view. To select a database view (srs or srm) with find in batch mode, use open_design (open_design, on page 50).</p>

Argument	Description
-print	<p>Prints the first 20 results. For a full list of objects found, use <code>c_print</code> or <code>c_list</code>.</p> <p>If you specify this command from an HDL Analyst view, the results are printed to the Tcl window; if you specify it in the constraint file, the results are printed to the log file, at the beginning of the Mapper section.</p> <p>Reported object names have prefixes that identify the object type, and double quotes around each name to allow for spaces in the names. For example:</p> <pre>"i:reg1" "i:\weird_name[foo\$] " "i:reg2"</pre> <p><<found 233 objects. Displaying first 20 objects. Use <code>c_print</code> or <code>c_list</code> for all. >></p>
-regexp	<p>Treats the pattern as a regular expression instead of a simple wildcard pattern. It also uses the <code>==</code> and <code>!=</code> filter operators to compare regular expressions, rather than simple wildcard patterns. You cannot use this argument with <code>-nocase</code> or <code>-exact</code>.</p> <p>If you do not specify <code>-regexp</code>, there are only two special characters that are used as wildcards: <code>*</code> and <code>?</code>. See Regular Expressions, Wildcards, and Special Characters, on page 106 for details about regular expressions.</p>
-rtl -tech	<p>Uses the most recently activated RTL or Technology view. If none are available, it opens a new view. The RTL view is the default.</p>
-seq	<p>Finds sequential (clocked) instances (the <code>-inst</code> object type is not required). This argument is equivalent to <code>-filter @is_sequential</code>.</p>

Tcl Find Command Object Types

You can specify the following types of objects:

Object	Prefix	Example	Synopsys
view (Design)	v:	v:work.cpu.rtl is the master cell of the cpu entity, rtl architecture, compiled in the VHDL work library.	lib_cell
inst (Instance)	i:	Default object type. i:core.i_cpu.reg1 points to the reg1 instance inside i_cpu.	cell
port	p:	p:data_in[3] points to bit 3 of the primary data_in port. work.cpu.rtl p:rst is the hierarchical rst port in the cpu view. This eventually points to all instances of cpu.	port
pin	t:	t:core.i_cpu.rst points to the hierarchical rst pin of instance i_cpu.	pin
net	n:	n:core.i_cpu.rst points to the rst net driven in i_cpu.	net
seq (Sequential instance)	i:	i:core.i_cpu.reg[7:0]	cell

Regular Expressions, Wildcards, and Special Characters

The Tcl find command significantly differs from a simple Tcl search. A simple Tcl search does not treat any character, except for the backslash (\), as a special character, so * matches everything in a string. The Tcl find command uses various regular expressions and special characters, as shown in the following table.

Use curly brackets {} or double quotes to prevent the interpretation of special characters within a pattern, and the backslash to escape a single character.

Syntax Matches ...

Meta Characters: Used to match certain conditions in a string

^	At the beginning of the string
\$	At the end of the string. Use curly brackets {} or double quotes to prevent the interpretation of special characters within a pattern.

Syntax Matches ...

.	Any character. If you want to use the dot (.) as a hierarchy delimiter, you must escape it with a backslash (\), because it has a special meaning in regular expressions.
\k	Interprets and matches the specified non-alphanumeric character as an ordinary, non-reserved character (where <i>k</i> is the non-alphanumeric character). For example, \\$ matches a dollar symbol, not the character in its reserved sense of matching the end of a string. Similarly \.d in a.b.c.d.e indicates that c.d must be interpreted as part of the instance name, not as a hierarchy separator.
\c	The specified non-alphanumeric character (where <i>c</i> is the non-alphanumeric character) when it is used in an escape sequence.
	Equivalent to an OR.

Character Class: A list of characters to match

[<i>list</i>]	Any single character from the <i>list</i> . For example, [abc] matches a lower-case a, b, or c. To specify a range of characters in the list, use a dash. For example, [A-Za-z] matches any alphabetical character. Use curly brackets {} or double quotes to prevent the interpretation of special characters within a pattern.
[^ <i>list</i>]	Characters not in the list. You can specify a range, as described above. For example, [^0-9] matches any non-numeric character.
\	Used as a prefix to escape special characters like the following: ^ \$ \ . () [] { } ? + *

Escape Sequences: Shortcuts for common character classes

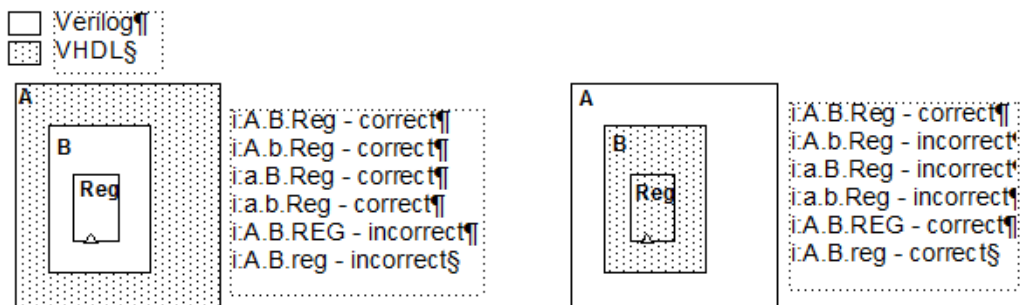
\d	A digit between 0 and 9
\D	A non-numeric character
\s	A white space character
\S	A non-white space character
\w	A word character; i.e., alphanumeric characters or underscores
\W	A non-word character

Syntax Matches ...**Quantifiers: Number of times to match the preceding pattern**

*	A sequence of 0 or more matches If you do not specify -hier, the search is restricted to the current view only. To traverse downward through the hierarchy, either use the -hier argument or specify the hierarchical levels to be searched by adding the hierarchical delimiter to the pattern. For example, *.* matches objects one level below the current view.
+	A sequence of 1 or more matches
?	A sequence of 0 or 1 matches
{N}	A sequence of exactly N matches Use curly brackets to interpret special characters as ordinary characters within a pattern.
{N,}	A sequence of N or more matches
{N,P}	A sequence of N through P matches (P included); $N \leq P$

Tcl Find Command Case Sensitivity

Case sensitivity depends on the rules of the language used to specify the object. If the object was generated in VHDL, it is case-insensitive; if it was generated in Verilog, it is case-sensitive. In mixed-language designs, the case-sensitivity rules for the parent object prevail, even when another language is used to define the lower-level object.



Tcl Find Syntax (Beta)

Beta

Finds design objects based on specified criteria.

Find (Beta) is available as part of HDL Analyst (Beta).

Syntax

```
find
    [-flat]
    [-inst]
    [-net]
    [-port]
    [-pin]
    [-view]
    [-nocase]
    [-print]
    [-depth value]
    [-filter expression]
    [-seq]
    [pattern]
```

-flat

Extends the search to all levels. The * wildcard character matches hierarchy separators as well as characters. See [Regular Expressions, Wildcards, and Special Characters, on page 106](#) for additional information.

-inst

Finds matching instances. If no -type (-inst, -net, -port, -pin, or -view) option is set, results include instances, nets, and ports.

-net

Finds matching nets. If no -type (-inst, -net, -port, -pin, or -view) option is set, results include instances, nets, and ports.

-port

Finds matching ports. If no -type (-inst, -net, -port, -pin, or -view) option is set, results include instances, nets, and ports.

-pin

Finds matching pins. If no -type (-inst, -net, -port, -pin, or -view) option is set, results include instances, nets, and ports.

-view

Finds matching views. If no **-type** (**-inst**, **-net**, **-port**, **-pin**, or **-view**) option is set, results include instances, nets, and ports.

-nocase

The **-nocase** option makes the search case-insensitive.

-print

Prints the first 20 search results. For a full list of objects found, use **c_print** or **c_list**. If you use **find** from the shell, the results are printed to the Tcl window; if you **find** in the constraint file, the results are printed to the log file at the beginning of the Mapper section. Reported object names have prefixes that identify the object type and are contained in curly braces ({ }).

-depth *value*

Sets the starting depth for the search. Value may be a single number or a range. When **-depth** with a range is used, for example **-depth 4-7**, **-hier** and **-flat** arguments are ignored.

-filter *expression*

Further refines the results of **find** by filtering the results using the specified object property. For syntax details, refer to [find -filter, on page 116](#).

-seq

Finds sequential (clocked) instances (the **-inst** object type is not required). This argument is equivalent to **-filter @is_sequential**.

pattern

The value to search for.

Tcl Find Command Object Types (Beta)

You can specify the following types of objects:

Object	Prefix	Example	Synopsys
view (Design)	v:	work.cpu.rt1 p:rst is the hierarchical rst port in the cpu view which points to all instances of cpu.	lib_cell
inst (Instance)	i:	Default object type. i:core.i_cpu.reg1 points to the reg1 instance inside i_cpu.	cell
port	p:	p:data_in[3] points to bit 3 of the primary data_in port. work.cpu.rt1 p:rst is the hierarchical rst port in the cpu view which eventually points to all instances of cpu.	port
pin	t:	t:core.i_cpu.rst points to the hierarchical rst pin of instance i_cpu.	pin
net	n:	n:core.i_cpu.rst points to the rst net driven in i_cpu.	net
seq (Sequential instance)	i:	i:core.i_cpu.reg[7:0]	cell

Wildcards and Special Characters (Beta)

The Tcl find command significantly differs from a simple Tcl search. A simple Tcl search does not treat any character, except for the backslash (\), as a special character, so * matches everything in a string. The Tcl find command uses various special characters, as shown in the following table.

Use curly brackets {} or double quotes to prevent the interpretation of special characters within a pattern, and the backslash to escape a single character.

Syntax Matches ...

*	A sequence of 0 or more matches If you do not specify -hier, the search is restricted to the current view only. To traverse downward through the hierarchy, either use the -hier argument or specify the hierarchical levels to be searched by adding the hierarchical delimiter to the pattern. For example, *.* matches objects one level below the current view.
?	A sequence of 0 or 1 matches

Tcl Find Command Case Sensitivity (Beta)

Case sensitivity depends on the rules of the language used to specify the object. In mixed-language designs, the case-sensitivity rules for the parent object prevail, even when another language is used to define the lower-level object.

Tcl Find Syntax Examples

The following are examples of find syntax:

Example	Description
find {a*}	Finds any object in the current view that starts with a
find {a*} -hier -nocase	Finds any object that starts with a or A
find -net {*synp*} -hier	Finds any net the contains synp
find -seq * -filter {@clock==myclk}	Finds any register in the current view that is clocked by myclk
find -flat -seq {U1.*}	Finds all sequential elements at any hierarchical level under U1 (* matches hierarchy separator)
find -hier -flat -inst {i:A.B.C.*} -filter @view==ram*	Finds all RAM instances starting from a submodule and all lower hierarchical levels from A downwards
find -hier-seq {*} -filter @clock_enable==ena	Finds all registers enabled by the ena signal.

Example	Description
<code>find -hier-seq {*} -filter @slack <{-0.0}</code>	Finds all sequential elements with negative slack.
<code>find -hier-seq {*} -filter {@clock ==clk1}</code>	Finds all sequential elements within the clk1 clock domain
<code>find -hier-net {*} -filter {@fanout >20}</code>	Finds high fanout nets that drive more than 20.destinations
<code>find -hier-seq * -in \$all_inst_coll</code>	Finds sequential elements inside the all_inst_coll collection
<code>find -net -regexp {[a-b].*}</code>	Finds all nets in hierarchy a and b. This means {n:a.*} and {n:b.*}

Use the `{}` characters to protect patterns that contain `[]` from Tcl evaluation. For example, use the following command to find instance `reg[4]`:

```
find -inst {reg[4]}
```

Example: Custom Report Showing Paths with Negative Slack

Use the following commands:

```
open_design implementation_a/top.srm
set find_negslack[find -hier -seq -inst {*} -filter @slack <
    {-0.0}]
c_print -prop slack -prop view $find_negslack -file negslack.txt
```

The result of running these commands is a report called `negslack.txt`:

```
Object Name                                slack  view
{i:CPU_A_SOC.CPU.DATAPATH.GBR[0]} -3.264 "FDE"
{i:CPU_A_SOC.CPU.DATAPATH.GBR[1]} -3.158 "FDE"
{i:CPU_A_SOC.CPU.DATAPATH.GBR[2]} -3.091 "FDE"
```

Example: Custom Report for Negative Slack FFs in a Clock Domain

The following procedure steps through the commands used to find all negative slack flip-flops with a given clock domain:

1. Create a collection that contains all sequential elements with negative slack:

```
set negFF [find -tech -hier -seq {*} -filter @slack < {-0.0}]
```

2. Create a collection of all sequential elements within the clk clock domain

```
set clk1FF find -hier -seq * -filter {@clock==clk1}
```

3. Isolate the common elements in the two collections:

```
set clk1Slack [c_intersect $negFF $clk1FF]
```

4. Generate a report using the c_print command:

```
c_print [find -hier -net * -filter @fanout>=2]
{n:ack1_tmp}
{n:ack2_tmp}
...
{n:blk_xfer_cntrl_inst.lfsr_data[20:14]}
{n:blk_xfer_cntrl_inst.lfsr_inst.blk_size[6:0]}
{n:blk_xfer_cntrl_inst.lfsr_inst.clk_c}
...
```

Custom Fanout Report Example

The following command generates a fanout report:

```
% c_print -prop fanout [find -hier -net * -filter @fanout>=2]
```

This is an example of the report generated by the command:

Object Name	fanout
{n:ack1_tmp}	3
{n:ack2_tmp}	4
...	
{n:blk_xfer_cntrl_inst.lfsr_data[14]}	3
{n:blk_xfer_cntrl_inst.lfsr_data[15]}	3
{n:blk_xfer_cntrl_inst.lfsr_data[16]}	2
...	

You can add additional information to the report, by specifying more properties. For example:

```
% c_print -prop fanout [find -hier -net * -filter @fanout>=2] -prop pins
```

This command generates a report like the one shown below:

Object Name	Fanout	Pins
{n:ack1_tmp}	3	"t:word_xfer_cntrl_inst.ack1_tmp t:word_xfer_inst.ack1_tmp"
{n:ack2_tmp}	4	"t:blk_xfer_cntrl_inst.ack2_tmp t:blk_xfer_inst.ack2_tmp"
{n:adr_o_axb_1}	2	"t:blk_xfer_inst.adr_o_axb_1 t:adr_o_cry_1_0.S t:adr_o_s_1.LI"
{n:adr_o_axb_2}	2	"t:blk_xfer_inst.adr_o_axb_2 t:adr_o_cry_2_0.S t:adr_o_s_2.LI"
{n:adr_o_axb_3}	2	"t:blk_xfer_inst.adr_o_axb_3 t:adr_o_cry_3_0.S t:adr_o_s_3.LI"
{n:adr_o_axb_4}	2	"t:blk_xfer_inst.adr_o_axb_4 t:adr_o_cry_4_0.S t:adr_o_s_4.LI"
{n:adr_o_axb_5}	2	"t:blk_xfer_inst.adr_o_axb_5 t:adr_o_cry_5_0.S t:adr_o_s_5.LI"
{n:adr_o_axb_6}	2	"t:blk_xfer_inst.adr_o_axb_6 t:adr_o_cry_6_0.S t:adr_o_s_6.LI"
...		

To save the report as a file, use a command like this one:

```
c_print -prop fanout [find -hier -net * -filter @fanout>=2]
      -prop pins -file prop.txt
```

find -filter

The Tcl find command includes the optional -filter option, which provides a powerful way to further refine the results of the find command and filter objects based on properties. See the following for details about the find -filter command:

- [Find -filter Syntax, on page 116](#)
- [Find Filter Properties, on page 117](#)
- [Find Filter Examples, on page 121](#)

For the Tcl find command syntax, see

Find -filter Syntax

find *pattern other_args* **-filter** **[!]{@property_name operator value}**

! Optional character to specify the negative. Include the ! character if you are checking for the absence of a property; leave it out if you are checking for the presence of a property.

@property_name Property name to use for filtering. The name must be prefixed with the @ character. For example, if clock is the property name, specify {@clock==myclk}.

operator Evaluates and determines the property value used for the filter expression. For the operators you can use in the expressions, see [Filter Operators, on page 117](#).

value Property value for the property in the filter expression, when the property has a value. The value can either be an object name such as myclk in {@clock==myclk}, or a value, such as 60 in {@fanout>=60}.

When specified, the -filter option must be the last option specified for the find command.

Filter Operators

You can use the following relational operators with the `-filter` option:

<code>==</code>	Equal
<code>!=</code>	Not equal
<code>></code>	Greater than
<code><</code>	Less than
<code>>=</code>	Greater than or equal to
<code><=</code>	Less than or equal to
<code>=~</code>	Matches pattern
<code>!~</code>	Does not match pattern

You can use the following logical operators with the `-filter` option:

<code>&&</code>	And
<code> </code>	Or
<code>!</code>	Not

Find Filter Properties

The object properties are based on the design or constraint, and are used to qualify searches and build collections. To generate these properties, open Project->Implementation Options->Device and enable the Annotated Properties for Analyst check box. The properties display in the Tcl window when the RTL or Technology view is active. Some properties are only available in a certain view. The tool creates `.sap` and `.tap` files (design and timing properties, respectively) in the project folder.

The table below lists the common filter object properties. It does not include some vendor-specific properties. Use the table as a guide to filter the properties you want. Here is how to read the columns:

Property Name	Property Value	HDL View	Description
Common Properties			
type	view port net instance pin	All	Specifies the type of object to be filtered from the netlist find * -filter -object -print
View Properties			
compile_point	locked	Tech	Filters the view based on compile point properties find * -view -filter @compile_point==locked -print
is_black_box	1	All	Filters the black box view find * -view -filter @is_black_box==1 -print
is_verilog	0 1	All	Filters the Verilog based view find * -view -filter @is_verilog=={0 1} -print
is_vhdl	0 1	All	Filters the VHDL based view find * -view -filter @is_vhdl=={0 1} -print
orig_inst_of	<i>viewName</i>	RTL and Tech	Filters the view based on original instance find * -view -filter @orig_inst_of==viewName -print
syn_hier	remove flatten soft firm hard	Tech	Filters the view based on the syn_hier attribute value find * -view -filter @syn_hier==(remove flatten soft firm hard) -print

Property Name	Property Value	HDL View	Description
Port Properties			
direction	input output inout	All	Filters the port based on port direction find * -port -filter @direction=={input output inout} -print
fanout	<i>value</i>	All	Filters the port based on fanout value find * -port -filter @fanout==value -print
Instance Properties			
area	<i>areaValue</i>	Tech	
arrival_time	<i>value</i>	Tech	Corresponds to worst slack
async_reset	n : <i>netName</i>	All	
async_set	n : <i>netName</i>	All	
clock	<i>clockName</i>	All	Could be a list if there are multiple clocks
clock_edge	rise fall high low	All	Could be a list if there are multiple clocks
clock_enable	n : <i>netName</i>	All	Highest branch name in the hierarchy, and closest to the driver
compile_point	locked	Tech	Automatically inherited from its view
hier_rtl_name	<i>hierInstanceName</i>	All	
inout_pin_count	<i>value</i>	All	
input_pin_count	<i>value</i>	All	
inst_of	<i>viewName</i>	All	
is_black_box	1 (Property added)	All	Automatically inherited from its view

Property Name	Property Value	HDL View	Description
is_hierarchical	1 (Property added)	All	
is_sequential	1 (Property added)	All	
is_combinational	1 (Property added)	All	
is_pad	1 (Property added)	All	
is_tristate	1 (Property added)	All	
is_keepbuf	1 (Property added)	All	
is_clock_gating	1 (Property added)	All	
is_vhdl	0 1	All	Automatically inherited from its view
is_verilog	0 1	All	Automatically inherited from its view
kind	<i>primitive</i> For example: inv and dff mux statemachine ...)	All	Tech view contains vendor-specific primitives
location	(<i>x</i> , <i>y</i>)	Tech	Format can differ
name	<i>instanceName</i>	All	
orientation	N S E W	Tech	
output_pin_count	<i>value</i>	All	
pin_count	<i>value</i>	All	
placement_type	unplaced placed	All	
rtl_name	<i>nonhierInstanceName</i>	All	
slack	<i>value</i>	Tech	Worst slack of all arcs
slow	1	Tech	
sync_reset	n : <i>netName</i>	All	
sync_set	n : <i>netName</i>	All	
syn_hier	remove flatten soft firm hard	Tech	Automatically inherited from its view
view	<i>viewName</i>	All	

Property Name	Property Value	HDL View	Description
Pin Properties			
arrival_time	<i>timingValue</i>	Tech	
clock	<i>clockName</i>	All	Could be a list if there are multiple clocks
clock_edge	rise fall high low	All	Could be a list if there are multiple clocks
direction	input output inout	All	
fanout	<i>value</i>	All	Total fanout (integer)
is_clock	0 1	All	
slack	<i>value</i>	Tech	
Net Properties			
clock	<i>clockName</i>	All	Could be a list if there are multiple clocks
is_clock	0 1	All	
fanout	<i>value</i>	All	Total fanout (integer)

Find Filter Examples

The following examples show how `find -filter` is used to check for the presence or absence of a property, with the `!` character indicating a negative check:

<code>c_print [find -hier -view {*} -filter (@is_black_box)]</code>	Finds all objects that are black boxes.
<code>c_print [find -hier -view {*} -filter (!@is_black_box)]</code>	Finds all objects that are not black boxes

The following are additional positive check examples:

```
find * -filter @fanout>8
(Finds all ports, pins, and nets from the top level with a fanout greater than 8)
```

```
find -pin *.* -filter @const
```

(Finds all constants driving the module pins preserved for constant propagation)

```
find * -hier -filter @view!=andv || @view!=orv
```

(Finds all instances other than andv and orv in the design)

```
find -hier -inst * -filter @inst_of==statemachine
```

(Finds all instances of statemachine throughout the hierarchy)

```
find -hier -inst * -filter @kind==statemachine
```

(Finds all instances of statemachine throughout the hierarchy)

```
find -hier -inst {*reg*} -filter @clock==CLK
```

(Finds all instances throughout the hierarchy with the name reg and that are clocked by CLK)

```
find -hier -net {*} -filter (@fanout > 4)
```

(Finds all nets throughout the hierarchy that have a fanout greater than 4)

This is another example of a negative check:

```
find -inst *big* -filter (!@is_black_box && @pin_count > 10
```

(Finds all instances from the top level that have the name big, are not black boxes, and have more than 10 pins)

You can also specify Boolean expressions on multiple properties:

```
find * -filter @pin_count>8 && @slack<0
```

(Finds all instances from the top level that have more than 8 pins and with negative slack)

expand

The `expand` command identifies objects based on their connectivity, by expanding forward from a given starting point. For more information, see [Using the Tcl expand Command to Define Collections, on page 143](#) of the *User Guide*.

Tcl expand Syntax

The syntax for the `expand` command is as follows:

```
expand [-objectType] [-from object] [-thru object] [-to object] [-level integer]
        [-hier] [-leaf] [-seq] [-print]
```

Argument	Description
-from <i>object</i>	Specifies a list or collection of ports, instances, pins, or nets for expansion forward from all the pins listed. Instances and input pins are automatically expanded to all output pins of the instances. Nets are expanded to all output pins connected to the net. If you do not specify this argument, backward propagation stops at all sequential elements.
-hier	Searches for the pattern from every level of hierarchy, instead of just the top level and identifies objects to be expanded based on their connectivity. The default for the current view is the top level and is defined with the <code>define_current_design</code> command as in the compile-point flow.
-leaf	Returns only non-hierarchical instances.
-level <i>integer</i>	Limits the expansion to N logic levels of propagation. You cannot specify more than one <code>-from</code> , <code>-thru</code> , or <code>-to</code> point when using this option.
-objectType	Optionally specifies the type of object to be returned by the expansion. If you do not specify an <i>objectType</i> , all objects are returned. The object type is one of the following: <ul style="list-style-type: none"> • -instance returns all instances between the expansion points. This is the default. • -pin returns all instance pins between the expansion points. • -net returns all nets between the expansion points. • -port returns all top-level ports between the expansion points.

Argument	Description
-print	<p>Evaluates the expand function and prints the first 20 results. If you use this command from HDL Analyst, these results are printed to the Tcl window; for constraint file commands, the results are printed to the log file at the start of the Mapper section.</p> <p>For a full list of objects found, you must use <code>c_print</code> or <code>c_list</code>. Reported object names have prefixes that identify the object type. There are double quotes around each name to allow for spaces in the names. For example:</p> <pre>"i:reg1" "i:reg2" "i:\weird_name[foo\$] " "i:reg3"</pre> <p><<found 233 objects. Displaying first 20 objects. Use <code>c_print</code> or <code>c_list</code> for all. >></p>
-seq	<p>Modifies the range of any expansion to include only sequential elements. By default, the expand command returns all object types. If you want just sequential instances, make sure to define the <i>object_type</i> with the <code>-inst</code> argument, so that you limit the command to just instances.</p>
-thru object	<p>Specifies a list or collection of instances, pins, or nets for expansion forward or backward from all listed output pins and input pins respectively. Instances are automatically expanded to all input/output pins of the instances. Nets are expanded to all input/output pins connected to the net. You can have multiple <code>-thru</code> lists for product of sum (POS) operations.</p>
-to object	<p>Specifies a list or collection of ports, instances, pins, or nets for expansion backward from all the pins listed. Instances and output pins are automatically expanded to all input pins of the instances. Nets are expanded to all input pins connected to the net.</p> <p>If you do not specify this argument, forward propagation stops at all sequential elements.</p>

Tcl expand Syntax Examples

Example	Description
<code>expand -hier -from {i:reg1} -to {i:reg2}</code>	Expands the cone of logic between two registers. Includes hierarchical instances below the current view.
<code>expand -inst -from {i:reg1}</code>	Expands the cone of logic from one register. Does not include instances below the current view.
<code>expand -inst -hier -to {i:reg1}</code>	Expands the cone of logic to one register. Includes hierarchical instances below the current view.
<code>expand -pin -from {t:i_and2.z} -level 1</code>	Finds all pins driven by the specified pin. Does not include pins below the current view.
<code>expand -hier -to {t:i_and2.a} -level 1</code>	Finds all instances driving an instance. Includes hierarchical instances below the current view.
<code>expand -hier -from {n:cen}</code>	Finds all elements in the transitive fanout of a clock enable net, across hierarchy.
<code>expand -hier -from {n:cen} -level 1</code>	Finds all elements directly connected to a clock enable net, across hierarchy.
<code>expand -hier -thru {n:cen}</code>	Finds all elements in the transitive fanout and transitive fanin of a clock enable net, across hierarchy.

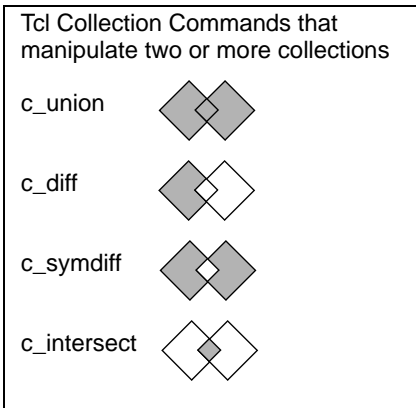
Collection Commands

A collection is a group of objects. Grouping objects lets you operate on multiple group members at once; for example you can apply the same constraint to all the objects in a collection. You can do this from both the SCOPE editor (see [Collections, on page 161](#)) or in a Tcl file.

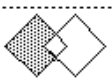
The following table lists the commands for creating, copying, evaluating, traversing, and filtering collections, and subsequent sections describe the collections, except for find and expand, in alphabetical order. For information on using collections, see [Using Collections, on page 148](#) in the *User Guide*.

Command	Description
Creation	
define_collection	Creates a collection from a list
set modules	Creates a collection
set modules_copy \$modules	Copies a collection
Creation from Objects Identified by Embedded Commands	
find	Does a targeted search and finds objects. Embedding the find command in a collection creation command first finds the objects, and then creates a collection out of the identified group of objects.
expand	Identifies related objects by expanding from a selected point. Embedding the expand command in a collection creation command first finds the objects, and then creates a collection out of the identified group of objects.
Operators for Comparison and Analysis	
c_diff	Identifies differences between lists or collections
c_intersect	Identifies objects common to a list and a collection
c_symdiff	Identifies objects that belong exclusively to only one list or collection
c_union	Concatenates a list to a collection

Command	Description
Operators for Evaluation and Statistics	
c_info	Prints statistics for a collection
c_list	Converts a collection to a Tcl list for evaluation
c_print	Displays collections or properties for evaluation



c_diff



Identifies differences by comparing collections, or a list and a collection. For this command to work, the design must be open in the GUI.

Syntax

```
c_diff {$collection1 $collection2 | $collection {list}} [-print]
```

This command also includes a -print option to display the result.

Examples

The following examples combine the `set` with the `c_diff` command to create a new collection that contains the results of the `c_diff` command. The first example compares two collections and puts the results in `diffCollection`:

```
set diffCollection [c_diff $collection1 $collection2]
```

The next example creates `collection1` consisting of objects `i:reg1` and `i:reg2`, compares this collection to a Tcl list containing object `i:reg1`, puts the results in the collection `diffCollection` and prints the result (`i:reg2`).

```
%set collection1 {i:reg1 i:reg2}
%set diffCollection [c_diff $collection1 {i:reg1}]
%c_print $diffCollection
{i:reg2}
```

c_info

Returns specifics of a collection, including database name, number of objects per type, and total number of objects. You can save the results to a Tcl variable (array) using the `-array name` option.

Syntax

```
c_info $mycollection [-array name]
```

c_intersect



Defines common objects that are included in each of the collections or lists being compared.

Syntax

```
c_intersect $collection1 $collection2 | list [-print]
```

This command also includes a `-print` option to display the result.

Example

The following example uses the `set` command to create a new collection that contains the results of the `c_intersect` command. The example compares a list to a collection (`myCollection`) and puts the common elements in a new collection called `commonCollection`:

```
%set myCollection {i:reg1 i:reg2}
%set commonCollection [c_intersect $myCollection {i:reg1 i:reg3}]
%c_print $commonCollection
    {i:reg1}
```

c_list

Converts a collection to a Tcl list of objects. You can evaluate any collection with this command. If you assign the collection to a variable, you can then manipulate the list using standard Tcl list commands like `lappend` and `lsort`. Optionally, you can specify object properties to add to the resulting list with the `-prop` option:

```
(object prop_value ... prop_value)...
    (object prop_value ... prop_value)
```

Syntax

c_list *\$collection|list* [-prop *propertyName*]*

Example

```
%set myModules [find -view *]
%c_list $myModules
{v:top}{v:block_a}{v:block_b}

%c_list $myModules -prop is_vhdl -prop is_verilog
```

Name	is_vhdl	is_verilog
{v:top}	0	1
{v:block_a}	1	0
{v:block_b}	1	0

c_print

Displays collections or properties in column format. Object properties are printed using one or more `-prop propertyName` options.

Syntax

```
c_print {$collection | {list}} [-prop propertyName]* [-file filename]
```

To print to a file, use the `-file` option. The following command in a constraint file prints the whole collection to a file:

```
c_print -file foo.txt $col
```

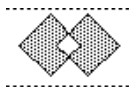
Note that the command prints the file to the current working directory. If you have multiple projects loaded, check that the file is written to the correct location. You can use the `pwd` command in the Tcl window to echo the current directory and then use `cd directoryName` to change the directory as needed.

Example

```
%set modules [find -view *]
%c_print $modules
{v:top}
{v:block_a}
{v:block_b}

%c_print -prop is_vhdl -prop is_verilog $modules
Name is_vhdl is_verilog
{v:top}0 1
{v:block_a}1 0
{v:block_b}1 0
```

c_symdiff



Compares a collection to another collection or Tcl list and finds the objects that are unique, not shared between the collections or Tcl lists being compared. It is the complement of the `c_intersect` command ([c_intersect](#), on [page 128](#)).

Syntax

```
c_symdiff {$collection1 $collection2 | $collection {list}} [-print]
```

This command also includes a -print option to display the result.

Examples

The following example uses the set command together with the c_symdiff command to compare two collections and create a new collection (symDiffCollection) that contains the results of the c_symdiff command.

```
set symDiff_collection [c_symdiff $collection1 $collection2]
```

The next example is more detailed. It compares a list to a collection (collection1) and creates a new collection called symDiffCollection from the objects that are different. In this case, reg1 is excluded from the new collection because it is common to both the list and collection1.

```
set collection1 {i:reg1 i:reg2}
set symDiffCollection [c_symdiff $collection1 {i:reg1 i:reg3}]
c_list $symDiffCollection
    {"i:reg2" "i:reg3"}
```

You can also use the command to compare two collections:

c_union



Adds a collection, or a list to a collection, and removes any redundant instances. For this command to work, the design must be open in the GUI.

Syntax

```
c_union {$collection1 $collection2 | $collection {list}} [-print]
```

The c_union command automatically removes redundant elements. This command also includes a -print option to display the result.

Examples

You can concatenate two collections into a new collection using the `c_union` and `set` commands, as shown in the following example where `collection1` and `collection2` are concatenated into `combined_collection`:

```
set combined_collection [c_union $collection1 $collection2]
```

The following example creates a new collection called `sumCollection`, which is generated by adding a Tcl list with one object (`reg3`) to `collection1`, which consists of `reg1` and `reg2`. The new collection created by `c_union` contains `reg 1`, `reg2`, and `reg3`.

```
%set collection1 [find -instance {reg?} -print]
    {i:reg1}
    {i:reg2}
%set sumcollection [c_union $collection1 {i:reg3}]
%c_list $sumcollection
    {i:reg1} {i:reg2} {i:reg3}
```

If instead you added `reg2` and `reg3` to `collection1` with the `c_union` command, the command removes redundant instances (`reg2`), so that the new collection still consists of `reg1`, `reg2`, and `reg3`.

```
%set collection1 {i:reg1 i:reg2}
%set sumcollection [c_union $collection1 {i:reg2 i:reg3}]
%c_list $sumcollection
    {i:reg1} {i:reg2} {i:reg3}
```

define_collection

Creates a collection from any combination of single elements, Tcl lists, and collections. You get a warning message about empty collections if you define a collection with a leading asterisk and then define an attribute for it, as shown here:

```
set noretimesh [define_collection [find -hier -seq *uc_alu]]
define_attribute {$noretimesh} {syn_allow_retiming} {0}
```

To avoid the error message, remove the leading asterisk and change `*uc_alu` to `uc_alu`.

Example

```
set modules [define_collection {v:top} {v:cpu} $mycoll $mylist]
```


define_scope_collection

The `define_scope_collection` command combines `set` and `define_collection` to create a collection and assigns it to a variable.

```
define_scope_collection my_regs {find -hier -seq *my*}
```

get_prop

Returns a single property value for each member of the collection in a Tcl list.

Examples

```
get_prop -prop clock [find -seq *]
get_prop $listExpandedInst -prop rtl_name LOROM32X1inst
get_prop $listExpandedInst -prop location SLICE_X1y36
get_prop $listExpandedInst -prop bel C6LUT
get_prop $listExpandedInst -prop slack 0.678
```

If this command is used in a Tcl script and the results need to be printed, use a puts command.

```
foreach cel [c_list $all_hier] {puts [get_prop -prop view $cel];}
```

set

Copies a collection to create a new collection. This command copies the collection but not the name, so the two are independent. Changes to the original collection do not affect the copied collection.

Syntax

set *collectionName* *collectionCriteria*

set *copyName* **\$***collectionName*

<i>collectionName</i>	The name of the new collection.
<i>collectionCriteria</i>	Criteria for defining the elements to be included in the collection. Use this argument to embed other commands, like Tcl find and expand, as shown in the examples below, or other collection commands like define_collection, c_intersect, c_diff, c_union, and c_symdiff. Refer to these commands for examples.
<i>copyName</i>	The name assigned to the copied collection.
\$ <i>collectionName</i>	Name of an existing collection to copy.

Examples

The following syntax examples illustrate how to use the set command:

- Use the set command to copy a collection:

```
set my_mod_copy $my_module
```

- Use the set command with a variable name and an embedded find command to create a collection from the find command results:

```
set my_module [find -view *]
```

- Use the set command with define_collection to create a collection:

```
set my_module [define_collection {v:top} {v:cpu} $col_1 $mylist]
```

For more examples of the set command used with embedded Tcl collection commands, see the examples in [c_diff, on page 127](#), [c_intersect, on page 128](#), [c_syndiff, on page 130](#), [c_union, on page 131](#), and [define_collection, on page 132](#).

Query Commands

The query commands are Synopsys SDC commands from the Design Compiler® tool for creating collections of specific object types. Functionally, they are equivalent to the Tcl find and expand commands ([find, on page 102](#) and [expand, on page 123](#)).

These query commands are intended to be used in the FDC file or the HDL Analyst view (see [Query Commands in HDL Analyst Tool, on page 137](#)) to create collections of objects for constraints. This section describes the syntax for the query commands supported in the FPGA synthesis tools. For complete documentation on these commands, see the Design Compiler documentation.

- [all_clocks](#)
- [all_fanin](#)
- [all_fanout](#)
- [all_inputs](#)
- [all_outputs](#)
- [all_registers](#)
- [get_cells](#)
- [get_clocks](#)
- [get_clock_source](#)
- [get_nets](#)
- [get_pins](#)
- [get_ports](#)
- [object_list](#)
- [report_timing](#)

Note: Since all the query commands above are used to create Tcl collections of objects for constraints, they must be enclosed in [] to be applied. For example:

```
set_input_delay 0.5 [all_inputs] -clock clk
```

Query Commands in HDL Analyst Tool

Most of the query commands can be used in both the FDC file and the HDL Analyst view to create collections of objects for constraints. However, the `all_clocks` command cannot be implemented in the HDL Analyst view. Note that the `get_clock_source` command is only used in the HDL Analyst view. This feature allows you to test the query commands in the HDL Analyst tool, before implementing them in the FDC file.

To use the query commands in the HDL Analyst RTL view:

1. Enable the option: Implementation Options->Device->Annotate Properties for Analyst.
2. If results are not as expected, check that this option is turned on during compile and before you open the SRS view.

Annotated Properties for Analyst	<input checked="" type="checkbox"/>
Verification Mode	<input type="checkbox"/>
Resolve Mixed Drivers	<input type="checkbox"/>
Read Write Check on RAM	<input checked="" type="checkbox"/>

Query Commands and Tcl find and expand Commands

The Synopsys `get*` commands and `all*` commands are functionally similar to the Tcl `find` and `expand` commands. The `get*` commands and `all*` commands are better suited to use with constraints and the `fdc` file, because they handle properties like `@clock` better than the Tcl `find` and `expand` commands. In certain cases, the `fdc` file does not support the `find` and `expand` commands, although you can still enter them in the Tcl window. See [Query Commands and Tcl find and expand Commands, on page 137](#) for examples.

Query and Tcl find/expand Examples

The following table lists parallel examples that compare how to use either the Tcl find/expand or the get/all commands to query design objects and set constraints.

Return the output pins of top-level registers clocked by clkb (e.g. inst1.inst2.my_reg.Q)

all_registers	FDC Constraint: <pre>set_multicycle_path {4} -from [all_registers -no_hierarchy -output_pins -clock [get_clocks {clk_b}]] set_multicycle_path {4} -from [get_pins -of_objects [get_cells * -filter {@clock == clk_b}] -filter {@name == Q}]</pre>
---------------	--

find	Tcl Window: <pre>% define_collection [regsub -all {i:([^\s]+)} [join [c_list [find -inst * -filter @clock == clkfx]]] {t:\1.Q}]</pre>
------	--

Return all registers in the design clocked by the rising edge of clock clkfx

all_registers	FDC Constraint: <pre>set_multicycle_path {3} -to [all_registers -cells -rise_clock [get_clocks {clkfx}]] set_multicycle_path {3} -to [get_cells -hier * -filter {@clock == clkfx && @clock_edge == rise}]</pre>
---------------	--

find	Tcl Window: <pre>find -hier -inst * -filter {@clock == clkfx && @clock_edge == rise}</pre>
------	---

Return clock pins of all registers clocked by the falling edge of clkfx

all_registers	FDC Constraint: <pre>set_multicycle_path {2} -from [all_registers -clock_pins -fall_clock [get_clocks {clkfx}]] set_multicycle_path {2} -from [get_pins -of_objects [get_cells -hier * -filter {@clock == clkfx && @clock_edge == fall}] -filter {@name == C}]</pre>
---------------	---

find	Tcl Window: <pre>% find -hier -inst * -filter {@clock == clkfx && @clock_edge == fall}</pre>
------	---

Return the E pins of all instances of dffre cells (e.g. inst1.inst2.my_reg.E)

get_pins	FDC Constraint: <pre>set_multicycle_path -to [get_pins -filter {@name == E} -of_objects [get_cells -hier * -filter {@inst_of == dffre}]]</pre>
----------	---

find	Tcl Window and FDC Constraint: <pre>% regsub -all {i:([^\s]+)} [join [c_list [find -hier -inst * -filter @inst_of == dffre]]] {t:\1.E}]</pre>
------	--

all_clocks

Use this command in the fdc constraint file to return a collection of objects. This command is not supported in the HDL Analyst view.

Returns a collection of clocks in the current design.

Syntax

This is the supported syntax for the all_clocks command:

all_clocks

This command has no arguments. All clocks must be defined in the design before using this command. To create clocks, you can use the create_clock command.

Example

The following constraint sets a multicycle path from all the starting points.

```
set_multicycle_path 3 -from [all_clocks]
```

all_fanin

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Reports pins, ports, or cells for the fanin of the specified sinks in the to list.

Syntax

This is the supported syntax for the all_fanin command:

```
all_fanin  
  [-break_on_bboxes]  
  [-endpoints_only]  
  [-exclude_bboxes]  
  [-flat]
```

```

[-levels integer]
[-only_cells]
[-startpoints_only]
-to listC
[-trace_arcs all | timing]

```

Arguments

-break_on_bboxes	Stops timing fanin from traversing on black boxes.
-endpoints_only	Returns only timing end points.
-exclude_bboxes	Excludes black boxes from the final result.
-flat	The fanin function operates in flat mode. This function can be specified in hierarchical (default) or flat mode. For hierarchical mode, only objects in the same hierarchy level as the current sink are returned. The pins within a level of hierarchy below the sink are traversed, but are not reported.
-levels <i>integer</i>	Stops traversal when the perimeter of the search <i>integer</i> hops is reached. For example, a level 2 hop traverses through two levels of combinational logic and stops, instead of hopping through all levels and stopping at the first sequential or port object. Counting is performed for the layers of cells that are equidistant from the sink.
-only_cells	Results include a set of all cells from the timing fanin for <i>listC</i> .
-startpoints_only	Returns only timing start points.
-to <i>listC</i>	<i>Required.</i> Reports a list of sink pins, ports, or nets in the design and the timing fanin of each sink in the <i>listC</i> Tcl list or collection specified. When you specify a net, effectively all drivers on the net are listed.
-trace_arcs all timing	Specifies the type of combinational arcs to trace while traversing the fanin. You can specify either: <ul style="list-style-type: none"> • all – Permits tracing of all combinational arcs. This is the default. • timing – Permits tracing of valid timing arcs only.

Examples

The following examples show the timing fanin of a port in the design.

```
all_fanin -to [get_ports y*]
    {t:y_obuf[4].O t:y_obuf[3].O t:y_obuf[0].O t:y_obuf[1].O
    t:y_obuf[2].O t:y_obuf[5].O t:y_obuf[6].O t:y_obuf[7].O t:GND.G
    t:moduley_inst.y_c[0] t:moduley_inst.y_c[1]
    t:moduley_inst.y_c[2]}
```

```
all_fanin -to [get_ports y*] -startpoints_only -flat
    {t:moduley_inst.q[2].Q t:moduley_inst.q[1].Q
    t:moduley_inst.q[0].Q}
```

```
all_fanin -to [get_ports y*] -startpoints_only -flat -only_cells
    {i:moduley_inst.q[0] i:moduley_inst.q[1] i:moduley_inst.q[2]}
```

all_fanout

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Returns a set of pins, ports, or cells for the fanout of the specified sources in the from list.

Syntax

This is the supported syntax for the all_fanout command:

```
all_fanout
    [-break_on_bboxes]
    -clock_tree / -from listC
    [-endpoints_only]
    [-exclude_bboxes]
    [-flat]
    [-levels integer]
    [-only_cells]
    [-trace_arcs all |timing]
```

Arguments

-break_on_bboxes	Stops timing fanout from traversing on black boxes.
-------------------------	---

-clock_tree	<p>Uses all clock source pins and/or ports in the design as its list of sources. Clock sources are specified with the <code>create_clock</code> command. If there are no clocks or clocks have no sources, then the report is empty. The <code>-clock_tree</code> option generates a report displaying the clock trees or networks in the design.</p> <p>The <code>-clock_tree</code> and <code>-from</code> options are mutually exclusive.</p>
-endpoints_only	Returns only timing end points.
-exclude_bboxes	Excludes black boxes from the final result.
-flat	<p>The fanout function operates in flat mode.</p> <p>This function can be specified in hierarchical (default) or flat mode. For hierarchical mode, only objects in the same hierarchy level as the current source are returned. The pins within a level of hierarchy below the source are traversed, but are not reported.</p>
-from <i>listC</i>	<p>Specifies a list of source pins, ports, or nets in the design. The timing fanout for each source of the <i>listC</i> Tcl list or collection is reported. When you specify a net, effectively all load pins on the net are listed.</p> <p>The <code>-clock_tree</code> and <code>-from</code> options are mutually exclusive.</p>
-levels <i>integer</i>	Stops traversal when the perimeter of the search <i>integer</i> hops is reached. For example, a level 2 hop traverses through two levels of combinational logic and stops, instead of hopping through all levels and stopping at the first sequential or port object. Counting is performed for the layers of cells that are equidistant from the source.
-only_cells	Results include a set of all cells from the timing fanout for the <i>listC</i> .
-trace_arcs all timing	<p>Specifies the type of combinational arcs to trace while traversing the fanout. You can specify either:</p> <ul style="list-style-type: none"> • <code>all</code> – Permits tracing of all combinational arcs. This is the default. • <code>timing</code> – Permits tracing of valid timing arcs only.

Examples

The following examples show the timing fanout of a port in the design.

```

all_fanout -from [get_ports {a*}]
    {t:a_ibuf[0].I t:a_ibuf[1].I t:a_ibuf[2].I t:hold_a.D
    t:modulex_inst.a_c[0] t:modulex_inst.a_c[1]
    t:modulex_inst.a_c[2]}

all_fanout -from [get_ports {a*}] -level 1
    {t:a_ibuf[0].I t:a_ibuf[1].I t:a_ibuf[2].I}

all_fanout -from [get_ports {a*}] -flat -endpoints_only
    {t:hold_a.D t:modulex_inst.qa[0].D t:modulex_inst.qa[1].D
    t:modulex_inst.qa[2].D t:modulex_inst.qa_fast[0].D}

```

all_inputs

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Returns a collection of input or inout ports in the current design.

Syntax

This is the supported syntax for the all_inputs command:

```

all_inputs
    [-clock clockName]
    [-exclude_clock_port]

```

Arguments

-clock <i>clockName</i>	Limits the search to ports that have input delay relative to <i>clockName</i> .
-exclude_clock_port	Excludes clock ports from the search.

Examples

The following constraints set a default input delay.

```
set_input_delay 3 [all_inputs]
set_input_delay 3 -clock {clk} [all_inputs]
```

all_outputs

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Returns a collection of output or inout ports in the current design.

Syntax

This is the supported syntax for the all_outputs command:

```
all_outputs
  [-clock clockName]
```

Arguments

-clock <i>clockName</i>	Limits the search to ports that have output delay relative to <i>clockName</i> .
--------------------------------	--

Examples

The following constraints set a default output delay.

```
set_output_delay 2 [all_outputs]
set_output_delay 2 -clock {clk} [all_outputs]
```

all_registers

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Returns a collection of sequential cells or pins in the current design.

Syntax

This is the supported syntax for the `all_registers` command:

```
all_registers
  [-clock clockName]
  [-rise_clock clockName]
  [-fall_clock clockName]
  [-cells]
  [-data_pins]
  [-clock_pins]
  [-output_pins]
  [-no_hierarchy]
```

Arguments

-clock <i>clockName</i>	Searches only sequential cells that are clocked by the specified clock. By default, all sequential cells in the current design are searched.
-rise_clock <i>clockName</i>	Searches only sequential cells triggered by the rising edge of the specified clock. By default, all sequential cells in the current design are searched.
-fall_clock <i>clockName</i>	Searches only sequential cells triggered by the falling edge of the specified clock. By default, all sequential cells in the current design are searched.
-cells	Returns a collection of sequential cells that meet the search criteria. If you do not specify any of the object types, the command returns a collection of sequential cells.
-data_pins	Returns a collection of data pins for the sequential cells that meet the search criteria.
-clock_pins	Returns a collection of clock pins for the sequential cells that meet the search criteria.
-output_pins	Returns a collection of output pins for the sequential cells that meet the search criteria.

-no_hierarchy Limits the search to only the current level of hierarchy. Sub-designs are not searched.
By default, the entire hierarchy is searched.

Example

The following constraint sets a max delay target for timing paths leading to all registers.

```
set_max_delay 10.0 -to [all_registers]
```

The following constraint sets a max delay target for timing paths leading to all registers clocked by PHI2.

```
set_max_delay 10.0 -to [all_registers -clock [get_clocks PHI2]]
```

get_cells

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Creates a collection of cells from the current design that is relative to the current instance.

Syntax

This is the supported syntax for the `get_cells` command:

```
get_cells  
  [-hierarchical]  
  [-nocase]  
  [-regexp]  
  [-filter expression]  
  [pattern]
```

Arguments

-hierarchical Searches each level of hierarchy for cells in the design relative to the current instance. The object name at a particular level must match the patterns. For the cell `block1/adder`, a hierarchical search uses "adder" to find this cell name.
By default, the search is *not* hierarchical.

-nocase	Ensures that matches are case-insensitive. This applies for both the patterns argument and the filter operators (=~ and !~).
-regexp	Views the patterns argument as a regular expression rather than a simple wildcard pattern. The behavior of the filter operators (=~ and !~) have also been modified to use regular expression rather than simple wildcard patterns. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Rigidly quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression assumes matching begins at the beginning of the object name and ends matching at the end of an object name. You can expand the search by adding ". *" to the beginning or end of the expressions, as needed.
-filter expressions	Filters the collection with the specified expression. For each cell in the collection, the expression is evaluated based on the cell's attributes. If the expression evaluates to true, the cell is included in the result.
pattern	Creates a collection of cells whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. Pattern matching is case sensitive unless you use the -nocase option.

Examples

The following example creates a collection of cells that begin with o and reference an FD2 library cell.

```
get_cells "o*" -filter "@ref_name =~ FD2"
```

The following example creates a collection of cells connected to a collection of pins.

```
set pinsel [get_pins o*/cp]
get_cells -of_objects $pinsel
```

The following example creates a collection of cells connected to a collection of nets.

```
set netsel [get_nets tmp]
get_cells -of_objects $netsel
```

This example creates a collection of cells with the string BSCAN in its name. Make sure to use the "=~" operator when performing wildcard matching.

```
get_cells -hier * -filter {@hier_rtl_name =~ *BSCAN*}
```

get_clocks

Use this command in the fdc constraint file and in the HDL Analyst view (on a limited basis) to return a collection of objects.

Creates a collection of clocks from the current design.

Syntax

This is the supported syntax for the `get_clocks` command:

```
get_clocks
  [-nocase]
  [-regexp]
  [-filter expression]
  [pattern | -of_objects objects]
  [-include_generated_clocks]
```

Arguments

-nocase	Ensures that matches are case-insensitive. This applies for both the patterns argument and the filter operators (=~ and !~).
-regexp	Views the patterns argument as a regular expression rather than a simple wildcard pattern. The behavior of the filter operators (=~ and !~) have also been modified to use regular expression rather than simple wildcard patterns. When using the <code>-regexp</code> option, be careful how you quote the patterns argument and filter expression. Rigidly quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression assumes matching begins at the beginning of the object name and ends matching at the end of an object name. You can expand the search by adding ".*" to the beginning or end of the expressions, as needed.

-filter <i>expressions</i>	Filters the collection with the specified expression. For each clock in the collection, the expression is evaluated based on the clock's attributes. If the expression evaluates to true, the clock is included in the result. This option is not supported in the HDL Analyst view.
<i>pattern</i>	Creates a collection of clocks whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. Pattern matching is case sensitive unless you use the -nocase option.
-of_objects <i>objects</i>	Creates a collection of clocks that are defined for the given net or pin objects.
-include_generated_clocks	Creates a collection of clocks matching the search criteria and includes any clocks derived or generated from the source clocks found. This option is not supported in the HDL Analyst view.

Examples

The following example creates a collection of clocks that match the wildcard pattern.

```
get_clocks {*BUF_1*derived_clock*}
```

The following example creates a collection of clocks that match the given regular expression.

```
get_clocks -regexp {.*derived_clock}
```

The following example creates a collection that includes clka and any generated or derived clocks of clka.

```
get_clocks -include_generated_clocks {clka}
```

get_clock_source

Use this command to identify derived clock sources when debugging your design in the HDL Analyst view. **Syntax**

This is the supported syntax for the `get_clock_source` command:

```
get_clock_source clockAlias
```

Arguments

<i>clockAlias</i>	Specifies the clock alias name for the synthesis clock.
-------------------	---

get_nets

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Creates a collection of nets from the current design.

Syntax

This is the supported syntax for the `get_nets` command:

```
get_nets
    [-hierarchical]
    [-nocase]
    [-regexp | -exact]
    [-filter expression]
    [pattern | -of_objects objects]
```

Arguments

-hierarchical	Searches each level of hierarchy for nets in the design relative to the current instance. The object name at a particular level must match the patterns. For the net <code>block1/muxsel</code> a hierarchical search uses <code>muxsel</code> to find this net name. By default, the search is <i>not</i> hierarchical.
-nocase	Ensures that matches are case-insensitive. This applies for both the patterns argument and the filter operators (<code>=~</code> and <code>!~</code>).

-regexp	<p>Views the patterns argument as a regular expression rather than a simple wildcard pattern. The behavior of the filter operators (=~ and !~) have also been modified to use regular expression rather than simple wildcard patterns.</p> <p>When using the -regexp option, be careful how you quote the patterns argument and filter expression. Rigidly quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression assumes matching begins at the beginning of the object name and ends matching at the end of an object name. You can expand the search by adding ".*" to the beginning or end of the expressions, as needed.</p>
-filter expressions	<p>Filters the collection with the specified expression.</p> <p>For any nets in the collection, the expression is evaluated based on the net's attributes. If the expression evaluates to true, the net is included in the result.</p>
pattern	<p>Creates a collection of nets whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. Pattern matching is case sensitive unless you use the -nocase option.</p> <p>The patterns and -of_objects arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.</p>
-of_objects objects	<p>Creates a collection of nets connected to the specified objects. Each object can be a pin, port, or cell.</p>

Examples

The following example creates a collection of nets connected to a collection of pins.

```
set pinsel [get_pins {o_reg1.Q o_reg2.Q}]
get_nets -of_objects $pinsel
```

The following example creates a collection of nets connected to the E pin of any cell in the modulex_inst hierarchy.

```
get_nets {*.} -filter {@pins =~ modulex_inst.*.E}
```

get_pins

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Creates a collection of pins from the current design that match the specified criteria.

Syntax

This is the supported syntax for the get_pins command:

```
get_pins
  [-hierarchical]
  [-nocase]
  [-regexp | -exact]
  [-filter expression]
  [pattern] [-of_objects objects] [-leaf]
```

Arguments

-hierarchical	Searches each level of hierarchy for pins in the design relative to the current instance. The object name at a particular level must match the patterns. For the cell block1/adder/D[0], a hierarchical search uses adder/D[0] to find this pin name. By default, the search is <i>not</i> hierarchical.
-nocase	Ensures that matches are case-insensitive. This applies for both the patterns argument and the filter operators (=~ and !~).
-regexp	Views the patterns argument as a regular expression rather than a simple wildcard pattern. The behavior of the filter operators (=~ and !~) have also been modified to use regular expression rather than simple wildcard patterns. When using the -regexp option, be careful how you quote the patterns argument and filter expression. Rigidly quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression assumes matching begins at the beginning of the object name and ends matching at the end of an object name. You can expand the search by adding ".*" to the beginning or end of the expressions, as needed. The -regexp and -exact options are mutually exclusive; use only one.

-exact	<p>Treats wildcards as plain characters, so the meanings of these wildcard are not interpreted.</p> <p>The -regexp and -exact options are mutually exclusive; use only one.</p>
-filter expressions	<p>Filters the collection with the specified expression.</p> <p>For each pin in the collection, the expression is evaluated based on the pin's attributes. If the expression evaluates to true, the cell is included in the result.</p>
pattern	<p>Creates a collection of pins whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. Pattern matching is case sensitive unless you use the -nocase option.</p> <p>The patterns and -of_objects arguments are mutually exclusive; you can specify only one. If you do not specify any of these arguments, the command uses * (asterisk) as the default pattern.</p>
-of_objects objects	<p>Creates a collection of pins connected to the specified objects. Each object can be a cell or net.</p> <p>By default, the command considers only pins connected to the specified nets at the same hierarchical level. To consider only pins connected to leaf cells on the specified nets, use the -leaf option.</p> <p>You cannot use the -hierarchical option with the -of_objects option.</p>
-leaf	<p>Includes pins that are on leaf cells connected to the nets specified with the -of_objects option. The tool can cross hierarchical boundaries to find pins on leaf cells.</p>

Examples

The following example creates a collection for all pins in the design.

```
get_pins -hier *.*
```

The following example creates a collection for pins from the top-level hierarchy that match the regular expression.

```
get_pins -regexp {.*\.ena}
```

The following example creates a collection for pins throughout the hierarchy that match the regular expression.

```
get_pins -hier - regexp {.*\.ena}
```

The following example creates a collection of hierarchical pin names for the library cell pin DQSFOUND, for each instantiation of a library cell named PHASER_IN_PHY.

```
get_pins -filter {@name =~ DQSFOUND} -of_objects [get_cells -hier  
* -filter {@inst_of =~ PHASER_IN_PHY}]
```

The following example creates a collection of library cell pins with the string DRCK in its name, for each instantiation of a library cell with the string BSCAN in its name. Whenever you use wildcards to match names, make sure to specify the "=" operator instead of the "==" operator.

```
[get_pins -filter {@name=~*DRCK} -of_objects [get_cells -hier * -  
filter {@hier_rtl_name =~ *BSCAN*}]]
```

get_ports

Use this command in the fdc constraint file to return a collection of objects and/or in the HDL Analyst view to return a Tcl list of objects.

Creates a collection of ports from that match the specified criteria.

Syntax

This is the supported syntax for the `get_ports` command:

```
get_ports  
[-nocase]  
[-regexp]  
[-filter expression]  
[pattern]
```

Arguments

-nocase	Ensures that matches are case-insensitive. This applies for both the patterns argument and the filter operators (=~ and !~).
----------------	--

-regexp	<p>Views the patterns argument as a regular expression rather than a simple wildcard pattern. The behavior of the filter operators (=~ and !~) have also been modified to use regular expression rather than simple wildcard patterns.</p> <p>When using the -regexp option, be careful how you quote the patterns argument and filter expression. Rigidly quoting with curly braces around regular expressions is recommended. Regular expressions are always anchored; that is, the expression assumes matching begins at the beginning of the object name and ends matching at the end of an object name. You can expand the search by adding ".*" to the beginning or end of the expressions, as needed.</p>
-filter expressions	<p>Filters the collection with the specified expression.</p> <p>For each port in the collection, the expression is evaluated based on the port's attributes. If the expression evaluates to true, the port is included in the result.</p>
pattern	<p>Creates a collection of ports whose names match the specified patterns. Patterns can include the * (asterisk) and ? (question mark) wildcard characters. Pattern matching is case sensitive unless you use the -nocase option.</p> <p>The patterns and -of_objects arguments are mutually exclusive, so only specify one of them. If you do not specify either argument, the command uses * (asterisk) as the default pattern.</p>

Examples

The following example queries all input ports beginning with mode.

```
get_ports mode* -filter {@direction =~ input}
```

object_list

Translates object strings returned by query commands in the HDL Analyst tool to proper Tcl lists. This allows you to process the results using Tcl commands.

Syntax

This is the supported syntax for the object_list command:

```
object_list objectString
```

Arguments

<i>objectString</i>	Converts the object string returned by an FDC query command to proper Tcl lists.
---------------------	--

Examples

For example:

```
% foreach x [object_list [get_cells -hier {q[*]}]]
    {puts "Match: $x"}

Match: i:modulex_inst.q[7:0]
Match: i:moduley_inst\[4\].q[7:0]
```

report_timing

Alternatively, use this command to generate timing reports for a design from the Tcl window, which follows the standards set for the Design Compiler or PrimeTime® commands. One advantage this command has over the Timing Analyst GUI in the synthesis tool is that the filter options (for example, -from/-to/-through) support the FDC query commands.

Syntax

This is the supported syntax for the `report_timing` command:

```
report_timing
  [-delay_type max]
  [-fall_from clock]
  [-fall_to clock]
  [-file string]
  [-from list]
  [-max_paths int]
  [-nworst 1]
  [-path_type full]
  [-rise_from clock]
  [-rise_to clock]
  [-slack_margin float]
  [-through instance]
  [-to list]
```


Arguments

-delay_type <i>max</i>	Specifies the path type for the end points. This default value can be specified as <i>max</i> ; the maximum delay. All other values are ignored.
-fall_from <i>clock</i>	Reports paths from the falling edge of the specified clock. For a given clock, selects the starting points for paths clocked on the falling edge of the clock source.
-fall_to <i>clock</i>	Reports paths to the falling edge of the specified clock. For a given clock, selects the ending points for paths clocked on the falling edge of the clock source.
-file <i>string</i>	Allows the report to be re-directed to the specified file.
-from <i>list</i>	Reports paths from the specified port, register, register pin, or clock.
-max_paths <i>int</i>	Reports the number of paths to report for the path group. The default integer value is 1.
-nworst 1	Reports the maximum number of paths to report for each timing end point. The default is 1, which reports the single worst path at a given end point. All other values are ignored.
-path_type <i>full</i>	Specifies how to display the timing path. This default value can be specified as <i>full</i> . A complete timing report is displayed, for example, containing timing start and end points, required time, and slack. All other values are ignored.
-rise_from <i>clock</i>	Reports paths from the rising edge of the specified clock. For a given clock, selects the starting points for paths clocked on the rising edge of the clock source.
-rise_to <i>clock</i>	Reports paths to the rising edge of the specified clock. For a given clock, selects the ending points for paths clocked on the rising edge of the clock source.
-slack_margin <i>float</i>	Reports paths for the specified slack margin, which allows you to specify a floating point value for the range of worst slack times.
-through <i>instance</i>	Reports paths that pass through the specified pins or nets.
-to <i>list</i>	Reports paths to the specified ports, register, register pin, or clock.

Examples

The following example reports timing to all registers clocked by clkb.

```
%report_timing -to [all_registers -clock {clkb}]

##### START OF TIMING REPORT #####
# Timing Report written on Mon Dec 16 10:35:02 2013|
#

Top view:                top
Requested Frequency:      50.0 MHz
Wire load mode:           top
Paths requested:          1
to:                       moduley_inst.qa[7:0] moduley_inst.qb[7:0]

Worst From-To Path Information
*****

Path information for path number 1:
  Requested Period:                10.000
  - Setup time:                    -1.000
  + Clock delay at ending point:    0.909
  = Required time:                  11.909

  - Propagation time:              0.000
  = Slack :                        11.909

  Number of logic level(s):        1
  Starting point:                   c[7:0] / c[0]
  Ending point:                     moduley_inst.qa[0] / D
  The start point is clocked by     clkb [rising]
  The end point is clocked by       clkb [rising] on pin C
```

Instance/Net Name	Type	Pin Name	Pin Dir	Delay	Arrival Time	No. of Fan Out (s)
c[7:0]	Port	c[0]	In	0.000	0.000	-
c[0]	Net	-	-	0.000	-	1
c_ibuf[0]	IBUF	I	In	-	0.000	-
c_ibuf[0]	IBUF	O	Out	0.000	0.000	-
c_c[0]	Net	-	-	0.000	-	1
moduley_inst.qa[0]	FDE	D	In	-	0.000	-

Path delay compensated for clock skew. Clock skew is added to clock-to-out value, and is subtracted from setup time value

End clock path:

Instance/Net Name	Type	Pin Name	Pin Dir	Delay	Arrival Time	No. of Fan Out (s)
clkb	Port	c[0]	In	0.000	0.000	-
clkb	Net	-	-	0.000	-	1
clkb_IBUFG	IBUF	I	In	-	0.000	-
clkb_IBUFG	IBUF	O	Out	0.000	0.000	-
clkb_i	Net	-	-	0.000	-	1
moduley_inst.qa[0]	FDE	D	In	-	0.000	-

END OF TIMING REPORT #####]

Synopsys Standard Collection Commands

There are a number of Synopsys standard SDC collection commands that can be included in the fdc file. These commands are not compatible with the `define_scope_collection` command.

The collection commands let you manipulate or operate on multiple design objects simultaneously by creating, copying, evaluating, iterating, and filtering collections. This section describes the syntax for the following collection commands supported in the FPGA synthesis tools; for the complete syntax for these commands, refer to the Design Compiler documentation.

- [add_to_collection](#)
- [append_to_collection](#)
- [copy_collection](#)
- [foreach_in_collection](#)
- [get_object_name](#)
- [index_collection](#)
- [remove_from_collection](#)
- [sizeof_collection](#)

Use these commands in the FDC constraint file to facilitate the shared scripting of constraint specifications between the FPGA synthesis and Design Compiler tools.

add_to_collection

Adds objects to a collection that results in a new collection. The base collection remains unchanged. Any duplicate objects in the resulting collection are automatically removed from the collection.

Syntax

This is the supported syntax for the `add_to_collection` command:

```
add_to_collection  
    [collection1]  
    [objectSpec]
```

Arguments

<i>collection1</i>	Specifies the base collection to which objects are to be added. This collection is copied to a resulting collection, where objects matching <i>objectSpec</i> are added to this results collection.
<i>objectSpec</i>	Specifies a list of named objects or collections to add. Depending on the base collection type (heterogeneous or homogeneous), the searches and resulting collection may differ. For more information, see Heterogeneous Base Collection, on page 161 and Homogeneous Base Collection, on page 161 .

Description

The `add_to_collection` command allows you to add elements to a collection. The result is a new collection representing the objects added from the *objectSpec* list to the base collection. Objects are duplicated in the resulting collection, unless they are removed using the `-unique` option. If *objectSpec* is empty, then the new collection is a copy of the base collection. Depending on the base collection type (heterogeneous or homogeneous), the searches and resulting collection may differ.

Heterogeneous Base Collection

If the base collection is heterogeneous, then only collections are added to the resulting collection. All implicit elements of the *objectSpec* list are ignored.

Homogeneous Base Collection

If the base collection is homogeneous and any elements of *objectSpec* are not collections, then the command searches the design using the object class of the base collection.

When *collection1* is an empty collection, special rules apply to *objectSpec*. If *objectSpec* is not empty, at least one homogeneous collection must be in the *objectSpec* list (can be any position in the list). The first homogeneous collection in the *objectSpec* list becomes the base collection and sets the object class for the function.

Example

```
set result [get_cells{u*}]
get_object_name $result
```

```

==> {u:u1} {i:u2} {i:u3}

set result_1 [add_to_collection $result {get_cells {i:clkb_IBUFG}}]
get_object_name $result_1

==> {i:u1} {i:u2} {i:u3} {i:clkb_IBUFG}

```

See [append_to_collection](#).

append_to_collection

Adds objects to the collection specified by a variable, modifying its value. Objects must be unique, since duplicate objects are not supported.

Syntax

This is the supported syntax for the `append_to_collection` command:

```

append_to_collection
    [variableName]
    [objectSpec]

```

Arguments

<i>variableName</i>	Specifies a variable name. The objects matching <i>objectSpec</i> are added to the collection referenced by this variable.
<i>objectSpec</i>	Specifies a list of named objects or collections to add to the resulting collection.

Description

The `append_to_collection` command allows you to add elements to a collection. This command treats the *variableName* option as a collection, and appends all the elements of *objectSpec* to that collection. If the variable does not exist, it creates a collection with elements from the *objectSpec* as its value. So, a collection is created that was referenced initially by *variableName* or automatically if the *variableName* was not provided. However, if the variable exists but does not contain a collection, then an error is generated.

The `append_to_collection` command can be more efficient than the `add_to_collection` command ([add_to_collection](#), on page 160) when you are building a collection in a loop.

Example

```
set result [get_cells{u*}]
get_object_name $result

==> {u:u1} {i:u2} {i:u3}

append_to_collection result {get_cells {i:clkb_IBUFG}}
get_object_name $result

==> {i:u1} {i:u2} {i:u3} {i:clkb_IBUFG}
```

See [add_to_collection](#).

copy_collection

Duplicates the contents of a collection that results a new collection. The base collection remains unchanged.

Syntax

This is the supported syntax for the `copy_collection` command:

```
copy_collection
    [collection1]
```

Arguments

<i>collection1</i>	Specifies the collection to be copied.
--------------------	--

Description

The `copy_collection` command is an efficient mechanism to create a duplicate of an existing collection. It is sometimes more efficient and usually sufficient to simply have more than one variable referencing the same collection. However, whenever you want to copy the collection instead of reference it, use the `copy_collection` command.

Be aware that if an empty string is used for the *collection1* argument, the command returns an empty string. This means that a copy of the empty collection is an empty collection.

Example

```
set insts [define_collection {u1 u2 u3 u4}]
set result_copy [copy_collection $insts]
get_object_name $result_copy

==> {u1} {u2} {u3} {u4}
```

foreach_in_collection

Iterates on the elements of a collection.

Syntax

This is the supported syntax for the `foreach_in_collection` command:

```
foreach_in_collection
  [iterationVariable]
  [collections]
  [body]
```

Arguments

<i>iterationVariable</i>	Specifies the name of the iteration variable. It is set to a collection of one object. Any argument that accepts collections as an argument can also accept the <i>iterationVariable</i> , as they are the same data type.
<i>collections</i>	Specifies a list of collections on which to iterate.
<i>body</i>	Specifies a script to execute for the iteration. If the body of the iteration is modifying the netlist, all or part of the collection involved in the iteration can be deleted. The <code>foreach_in_collection</code> command is safe for such operations. A message is generated that indicates the iteration ended prematurely.

Description

The `foreach_in_collection` command is a Design Compiler and PrimeTime command used to iterate on each element of a collection. This command requires the following arguments: an iteration variable (do not specify a list), the collection on which to iterate, and the script to apply for each iteration.

You can nest this command within other control structures, including another `foreach_in_collection` command.

You can include the command in an FDC file, but if you are using the Tcl window and HDL Analyst, you must use the standard Tcl `foreach` command instead of `foreach_in_collection`.

Example

The following examples show valid methods to reference a collection for this command:

```
set seqs[all_registers]
set port[all_inputs]

foreach_in_collection x [all_registers] {body}
foreach_in_collection x $ports {body}
foreach_in_collection x [list $seqs $ports] {body}
foreach_in_collection x {$seqs} {body}
foreach_in_collection x {$seqs $ports} {body}
```

get_object_name

Returns a list of names for objects in a collection.

Syntax

This is the supported syntax for the `get_object_name` command:

```
get_object_name
    [collection1]
```

Arguments

<i>collection1</i>	Specifies the name of the collection that contains the requested objects.
--------------------	---

Example

```
set c1[define_collection {u1 u2}]
get_object_name $c1

==> {u1} {u2}
```

index_collection

Creates a new collection that contains only the single object for the index specified in the base collection. You must provide an index to the collection.

Syntax

This is the supported syntax for the `index_collection` command:

```
index_collection
  [collection1]
  [index]
```

Arguments

<i>collection1</i>	Specifies the collection to be searched.
<i>index</i>	Specifies an index to the collection. Allowed values are integers from 0 to <code>sizeof_collection - 1</code> .

Description

You can use the `index_collection` command to extract a single object from a collection. The result is a new collection that contains only this object. The range of indices can be from 0 to one less than the size of the collection. If the specified index is outside that range, an error message is generated.

Commands that create a collection of objects do not impose a specific order on the collection, but they do generate the objects in the same, predictable order each time. Applications that support the sorting of collections allow you to impose a specific order on a collection.

If you use an empty string for the *collection1* argument, then any index to the empty collection is not valid. This results in an empty collection and generates an error message.

Be aware that all collections cannot be indexed.

Example

```
set c1[get_cells {u1 u2}]]
get_object_name [index_collection $c1 0]

==> {u1}
```

See [sizeof_collection](#).

remove_from_collection

Removes objects from a collection that results in a new collection. The base collection remains unchanged.

Syntax

This is the supported syntax for the `remove_from_collection` command:

```
remove_from_collection
  [-intersect]
  [collection1]
  [objectSpec]
```

Arguments

-intersect	Removes objects from the base collection that are <i>not</i> found in <i>objectSpec</i> . By default, when this option is not specified, objects are removed from the base collection that are found in the <i>objectSpec</i> .
<i>collection1</i>	Specifies the base collection that is copied to a resulting collection, where objects matching <i>objectSpec</i> are removed from this results collection.
<i>objectSpec</i>	Specifies a list of named objects or collections to remove. The object class for each element in this list must be the same in the base collection. If the name matches an existing collection, that collection is used. Otherwise, objects are searched in the design using the object class for the base collection.

Description

The `remove_from_collection` command removes elements from a collection and creates a new collection.

When the `-intersect` option is not specified and there are no matches for *objectSpec*, the resulting collection is just a copy of the base collection. If everything in the *collection1* option matches the *objectSpec*, this results in an empty collection. When using the `-intersect` option, nothing is removed from the resulting collection.

Heterogeneous Base Collection

If the base collection is heterogeneous, then any elements of *objectSpec* that are not collections are ignored.

Homogeneous Base Collection

If the base collection is homogeneous and any elements of *objectSpec* are not collections, then the command searches the design using the object class of the base collection.

Examples

```
set c1[define_collection {u1 u2 u3}]
set c2[define_collection {u2 u3 u4}]
get_object_name [remove_from_collection $c1 $c2]

==> {u1}

get_object_name [remove_from_collection $c2 $c1]

==> {u4}

get_object_name [remove_from_collection -intersect $c1 $c2]

==> {u2} {u3}
```

See [add_to_collection](#).

sizeof_collection

Returns the number of objects in a collection.

Syntax

This is the supported syntax for the sizeof_collection command:

```
sizeof_collection  
    [collection1]
```

Arguments

<i>collection1</i>	Specifies the name of the collection for which the number of objects is requested. If no collection argument is specified, then the command returns 0.
--------------------	---

Examples

```
set c1[define_collection {u1 u2 u3}]  
sizeof_collection $c1  
  
==> 3
```


CHAPTER 4

User Interface Commands






The following describe the graphical user interface (GUI) commands available from the menus:

- [File Menu, on page 172](#)
- [Edit Menu, on page 177](#)
- [View Menu, on page 191](#)
- [Project Menu, on page 199](#)
- [Implementation Options Command, on page 214](#)
- [Import Menu, on page 243](#)
- [Run Menu, on page 244](#)
- [Analysis Menu, on page 291](#)
- [HDL Analyst Menu, on page 303](#)
- [Options Menu, on page 315](#)
- [Web Menu, on page 340](#)
- [Help Menu, on page 341](#)

For information about context-sensitive commands accessed from right-click popup menus, see [Chapter 5, GUI Popup Menu Commands](#).

File Menu

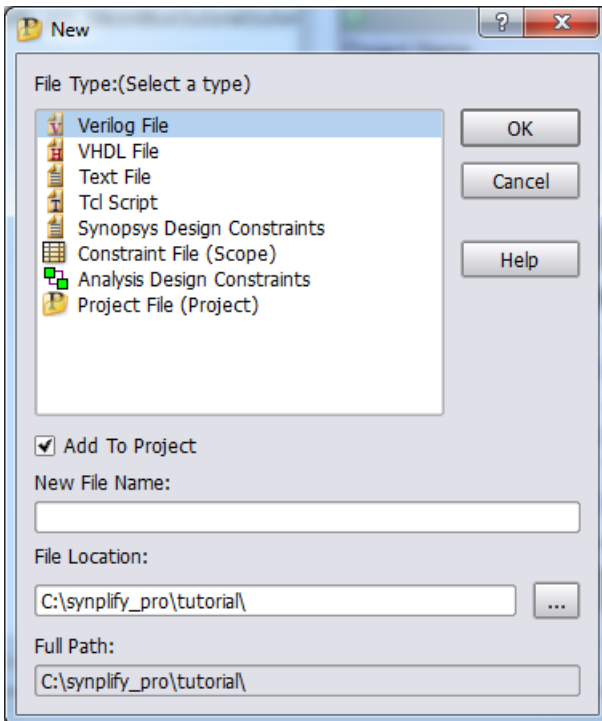
Use the File menu for opening, creating, saving, and closing projects and files. The following table describes the File menu commands.

Command	Description
New	Can create any of the following types of files: Text, Tcl Script, VHDL, Verilog, Design Constraints, Analysis Design Constraint, or Project file. See New Command, on page 173 .
 Open	Opens a project or file.
Close	Closes a project file.
 Save	Saves a project or a file.
Save As	Saves a project or a file to a specified name.
 Save All	Saves all projects or files.
Print	Prints a file. For more information about printing, see the operating system documentation.
Print Setup	Specify print options.
Create Image	<p>This command is available in the following views:</p> <ul style="list-style-type: none"> • HDL Analyst Views • FSM Viewer <p>A camera pointer () appears. Drag a selection rectangle around the region for which you want to create an image, then release the mouse button. You can also simply click in the current view, then the Create Image dialog appears. See Create Image Command, on page 174.</p>
Build Project	Creates a new project based on the file open in the Text Editor (if active), or lets you choose files to add to a new project. See Build Project Command, on page 176 .
 Open Project	Opens a project. See Open Project Command, on page 176 .
New Project	Creates a new project. If a project is already open, it prompts you to save it before creating a new one. If you want to open multiple projects, select Allow multiple projects to be opened in the Project View dialog box. See Project View Options Command, on page 319 .
Close Project	Closes the current project.

Command	Description
Recent Projects	Lists recently accessed projects. Choose a project listed in the submenu to open it.
Recent files (listed as separate menu items)	Lists the last files you opened recently as separate menu items. Choose a file to open it.
Exit	Exits the session.

New Command

Select File->New to display the New dialog box, where you can select a file type to be created (for example Verilog, VHDL, text, Tcl script, design constraints, analysis design constraints, or project). For most file types, a text editor window opens to allow you to define the file contents. You must provide a file name. You can automatically add the new file to your project by enabling the Add To Project checkbox before clicking OK.



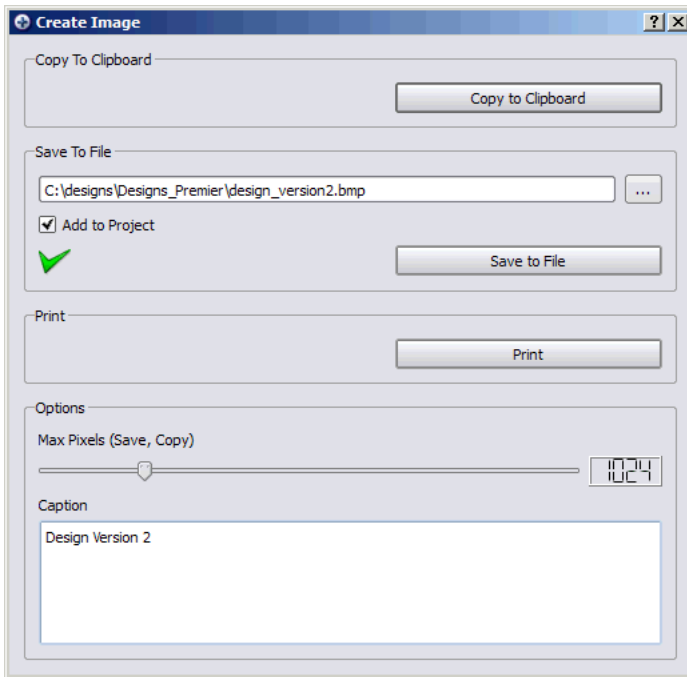
File Type	Opens Window	Directory Name	Extension
Verilog	Text Editor	Verilog	.v
VHDL	Text Editor	VHDL	.vhd
Text	Text Editor	Other	.txt
Tcl Script	Text Editor	Tcl Script	.tcl
FPGA Design Constraints	SCOPE	Constraint	.fdc
Analysis Design Constraints	SCOPE	Analysis Design Constraint	.adc
Project	None	None	.prj

Create Image Command

Select File->Create Image to create a capture image from any of the following views:

- HDL Analyst Views
- FSM Viewer

Drag the camera cursor to define the area for the image. When you release the cursor, the Create Image dialog box appears. Use the dialog box to copy the image, save the image to a file, or to print the image.



Field/Option	Description
Copy to Clipboard	Copies the image to the clipboard so you can paste it into a selected application (for example, a Microsoft Word file). When you copy an image to the clipboard, a green check mark appears in the Copy To Clipboard field.
Save to File	Saves the image to the specified file. You can save the file in a number of formats (platform dependent) including bmp, jpg, png, ppm, tif, xbm, and xpm.
Add to Project	Adds the saved image file to the Images folder in the Project view. This option is enabled by default.
Save to File button	You must click this button to save an image to the specified file. When you save the image, a green check mark appears in the Save To File field.

Field/Option	Description
Print	Prints the image. When you print the image, a green check mark appears in the Print field.
Options	Allows you to select the resolution of the image saved to a file or copied to the clipboard. Use the Max Pixels slider to change the image resolution.
Caption	Allows you to enter a caption for a saved or copied image. The overlay for the caption is at the top-left corner of the image.

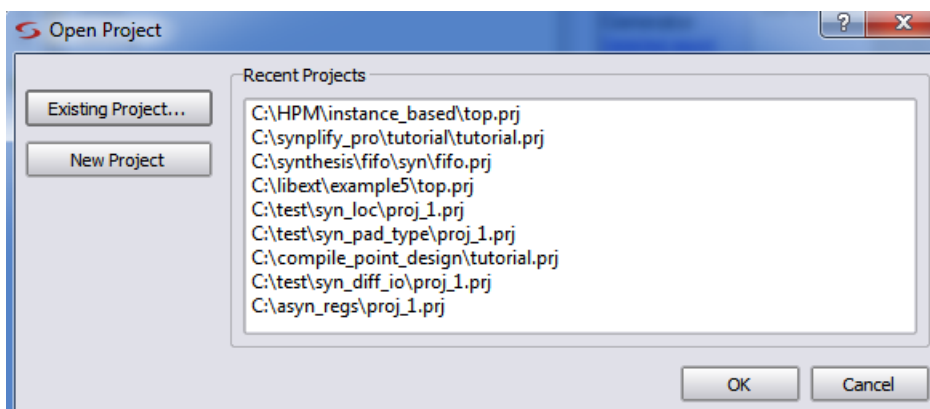
Build Project Command

Select File->Build Project to build a new project. This command behaves differently if an HDL file is open in the Text Editor.

- When an active Text Editor window with an HDL file is open, File->Build Project creates a project with the same name as the open file.
- If no file is open, File->Build Project prompts you to add files to the project using the Select Files to Add to New Project dialog box. The name of the new project is the name of the first HDL file added. See [Add Source File Command](#), on page 200.

Open Project Command

Select File->Open Project to open an existing project or to create a new project.













Field/Option	Description
Existing Project	Displays the Open Project dialog box for opening an existing project.
New Project	Creates a new project and places it in the Project view.

Edit Menu

You use the Edit menu to edit text files (such as HDL source files) in your project. This includes cutting, copying, pasting, finding, and replacing text; manipulating bookmarks; and commenting-out code lines. The Edit menu commands available at any time depend on the active window or view (Project, Text Editor, SCOPE spreadsheet, RTL, or Technology views).

The available Edit menu commands vary, depending on your current view. The following table describes all of the Edit menu commands:

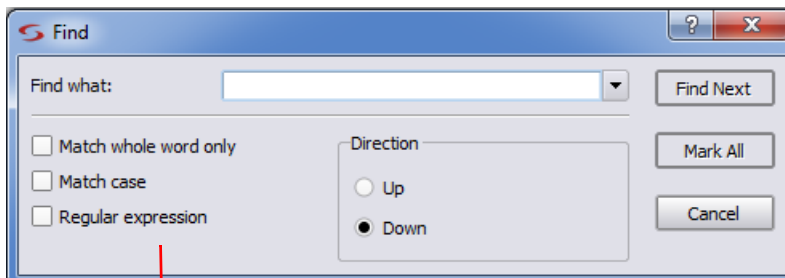
Command	Description
Basic Edit Menu Commands	
 Undo	Cancels the last action.
 Redo	Performs the action undone by Undo.
 Cut	Removes the selected text and makes it available to Paste.
 Copy	Duplicates the selected text and makes it available to Paste.
 Paste	Pastes text that was cut (Cut) or copied (Copy).
Delete	Deletes the selected text.
 Find	Searches the file for text matching a given search string; see Find Command (Text) , on page 179. In the RTL view, opens the Object Query dialog box, which lets you search your design for instances, symbols, nets, and ports, by name; see Find Command (HDL Analyst) , on page 183. In the project view, searches files for text strings; see Find Command (In Project) , on page 181.

Command	Description
Find Next	Continues the search initiated by the last Find.
Find in Files	Performs a string search of the target files. See Find in Files Command, on page 187 .
Edit Menu Commands for the Text Editor	
Select All	Selects all text in the file.
Replace	Finds and replaces text. See Replace Command, on page 189 .
Goto	Goes to a specific line number. See Goto Command, on page 190 .
 Toggle bookmark	Toggles between inserting and removing a bookmark on the line that contains the text cursor.
 Next bookmark	Takes you to the next bookmark.
 Previous bookmark	Takes you to the previous bookmark.
 Delete all bookmarks	Removes all bookmarks from the Text Editor window.
Advanced->Comment Code	Inserts the appropriate comment prefix at the current text cursor location.
Advanced -> Uncomment Code	Removes comment prefix at the current text cursor location.
Advanced->Uppercase	Makes the selected string all upper case.
Advanced->Lowercase	Makes the selected string all lower case.
Select->All	Selects all text in the file (same as All).

Find Command (Text)

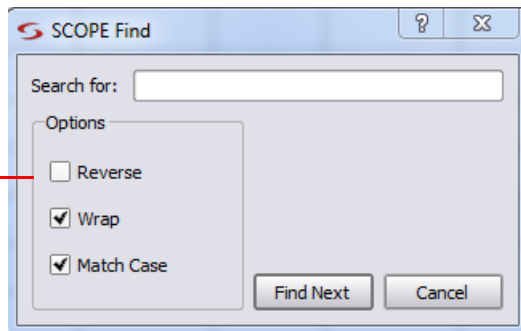
Select Edit->Find to display the Find dialog box. In the SCOPE window, the FSM Viewer, and the Text Editor window, the command has basic text-based search capabilities. Some search features, like regular expressions and line-number highlighting, are available only in the Text Editor. See [Find Command \(In Project\), on page 181](#), to search for files in the Project.

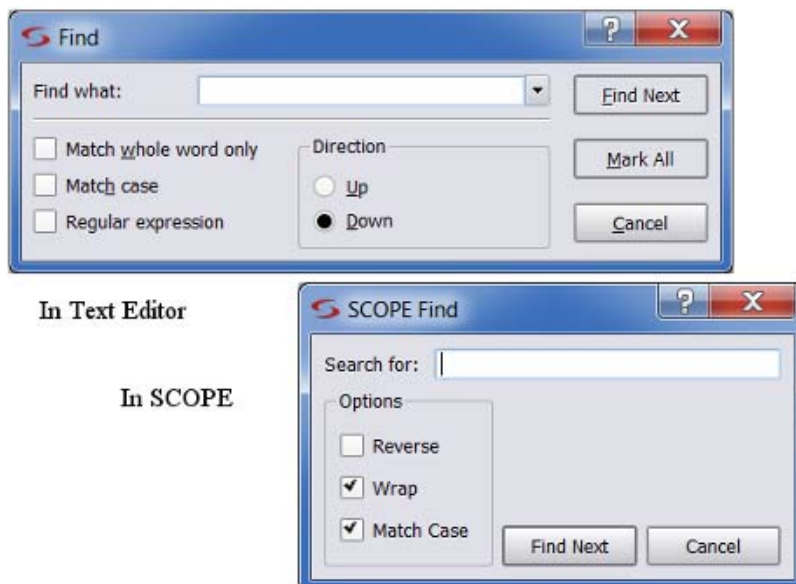
The HDL Analyst Find command is different; see [Find Command \(HDL Analyst\), on page 183](#) for details.



In Text Editor

In SCOPE





In Text Editor

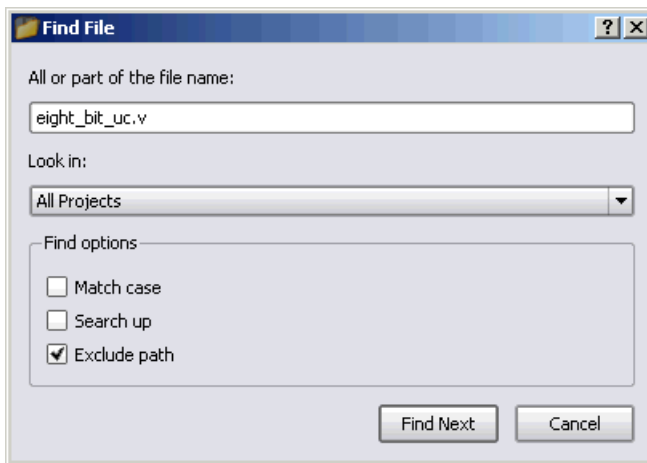
In SCOPE

Field/Option	Description
Find What/Search for	Search string matching the text to find. In the text editor, you can use the pull-down list to view and reuse search strings used previously in the current session.
Match whole word only (text editor only)	When enabled, matches the entire word rather than a portion of the word.
Match Case	When enabled, searching is case sensitive.
Regular expression (text editor only)	When enabled, wildcard characters (* and ?) can be used in the search string: ? matches any single character; * matches any string of characters, including an empty string.
Direction/Reverse	Changes search direction. In the text editor, buttons select the search direction (Up or Down).

Field/Option	Description
Find Next	Initiates a search for the search string (see Find What/Search for). In the text editor, searching starts again after reaching the end (Down) or beginning (Up) of the file.
Wrap (SCOPE only)	When enabled, searching starts again after reaching the end or beginning (Reverse) of the spread sheet.
Mark All (Text editor only)	Highlights the line numbers of the text matching the search string and closes the Find dialog box.

Find Command (In Project)

Select Edit->Find to display the Find File dialog box. In the Project view, the command has basic text-based search capabilities to locate files in the project.

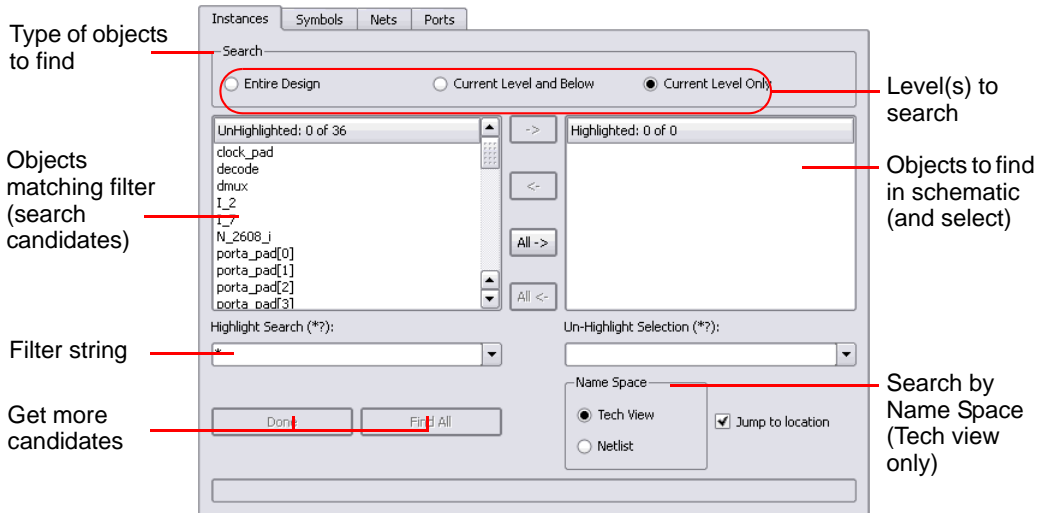


Field/Option	Description
All or part of the file name	Search string matching the file to find. You can specify all or part of the file name.
Look in	Search for files in all projects or limit the search to files only in the specified project.
Match Case	When enabled, searching is case sensitive.

Field/Option	Description
Search up	Searches in the up direction (search terminates when an end of tree is reached in either direction).
Exclude path	Excludes the path name during the search.
Find Next	Initiates a search for the file name string.

Find Command (HDL Analyst)

In the RTL or Technology view, use Edit->Find to display the Object Query dialog box. For a detailed procedure about using this command, see [Using Find for Hierarchical and Restricted Searches, on page 279](#) of the *User Guide*.



The available Find menu commands vary, depending on your current view. The following table describes all of the Find menu commands:

Field/Option	Description
Instances, Symbols, Nets, Ports	Tabbed panels for finding different kinds of objects. Choose a panel for a given object type by clicking its tab. In terms of memory consumption, searching for Instances is most efficient, and searching for Nets is least efficient.
Search	Where to search: Entire Design, Current Level & Below, or Current Level Only. See Using Find for Hierarchical and Restricted Searches, on page 279 of the <i>User Guide</i> .

Field/Option	Description
--------------	-------------

UnHighlighted	<p>Names of all objects of the current panel type, in the level(s) chosen to Search, that match the Highlight Search (*?) filter. This list is populated by the Find 200 and Find All buttons.</p> <p>To select an object as a candidate for highlighting, click its name in this list. The complete name of the selected object appears near the bottom of the dialog box. You can select part or all of this complete name, then use the Ctrl-C keyboard shortcut to copy it for pasting.</p> <p>You can select multiple objects by pressing the Ctrl or Shift key while clicking; press Ctrl and click a selection to deselect it. The number of objects selected, and the total number listed, are displayed above the list, after the UnHighlighted: label: # selected of # total.</p> <p>To confirm a selection for highlighting and move the selected objects to the Highlighted list, click the -> button.</p>
Highlight Search (*?)	<p>Determines which object names appear in the UnHighlighted area, based on the case-sensitive filter string that you enter. For tips about using this field, see Using Wildcards with the Find Command, on page 282 of the <i>User Guide</i>.</p> <p>The filter string can contain the following wildcard characters:</p> <ul style="list-style-type: none"> • * (asterisk) – matches any sequence of characters • ? (question mark) – matches any single character • . (period) – does not match any characters, but indicates a change in hierarchical level. <p>Wildcards * and ? only match characters within the current hierarchy level; a*b*, for example, will not cross levels to match alpha.beta (where the period indicates a change in hierarchy).</p> <p>If you must match a period character occurring in a name, use \. (backslash period) in the filter string. The backslash prevents interpreting the period as a wildcard.</p> <p>The filter string is matched at each searched level of the hierarchy (the Search levels are described above). Use filter strings that are as specific as possible to limit the number of unwanted matches. Unnecessarily extensive search can be costly in terms of memory performance.</p>
->	Moves the selected names from the UnHighlighted area to the Highlighted area, and highlights their objects in the RTL and Technology views.
<-	Moves the selected names from the Highlighted area to the UnHighlighted area, and unhighlights their objects in the RTL and Technology views.

Field/Option	Description
All ->	Moves all names from the UnHighlighted to the Highlighted area, and highlights their objects in the RTL and Technology views.
<- All	Moves all names from the Highlighted to the UnHighlighted area, and unhighlights their objects in the RTL and Technology views.
Highlighted	Complementary and analogous to the UnHighlighted area. You select object names here as candidates for moving to the UnHighlighted list. (You move names to the UnHighlighted list by clicking the <- button which unselects and unhighlights the corresponding objects.) When you select a name in the Highlighted list, the view is changed to show the (original, unfiltered) schematic sheet containing the object.
Un-Highlight Selection (*?)	Complementary and analogous to the Highlight Search area: selects names in the Highlighted area, based on the filter string you input here.
Jump to location	When enabled, jumps to another sheet if necessary to show target objects.
Name Space: Tech View	Searches for the specified name using the mapped (srn) database. For more information, see Using Find for Hierarchical and Restricted Searches, on page 279 of the <i>User Guide</i> .
Name Space: Netlist	Searches for the specified name using the output netlist file. For more information, see Using Find for Hierarchical and Restricted Searches, on page 279 of the <i>User Guide</i> .

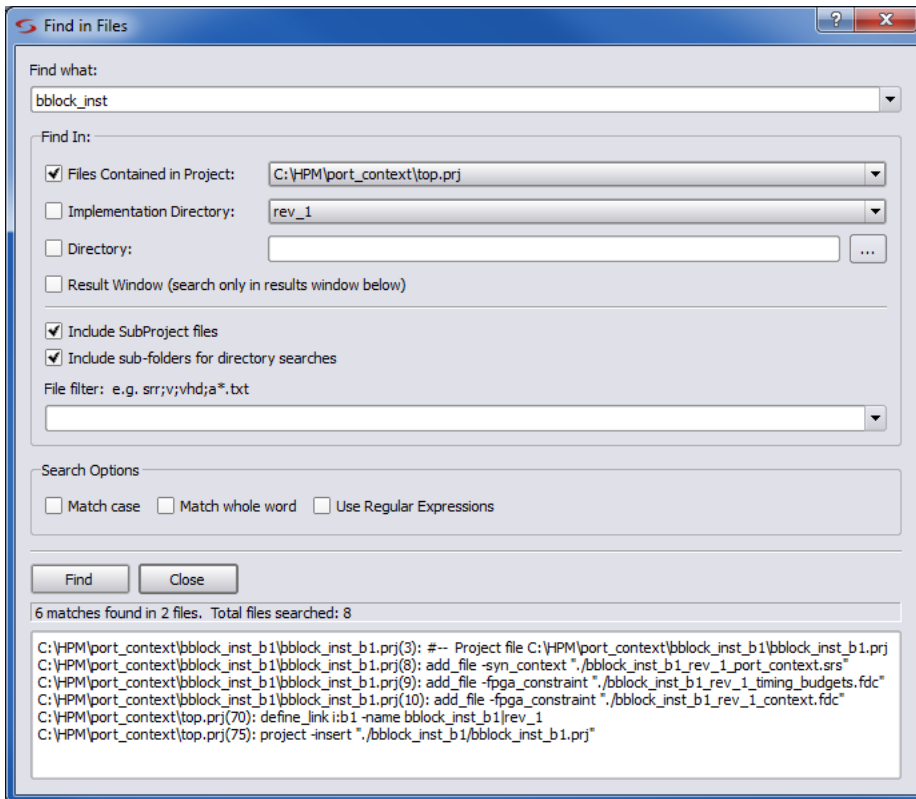
Field/Option	Description
--------------	-------------

Find 200	<p>Adds up to 200 more objects that match the filter string to the UnHighlighted list. This button becomes available after you enter a Highlight Search (*) filter string. This button does not find objects in HDL Analyst views. It matches names of design objects against the Highlight Search (*) filter and provides the candidates listed in the UnHighlighted list, from which you select the objects to find.</p> <p>Using the Enter (Return) key when the cursor is in the Highlight Search (*) field is equivalent to clicking the Find 200 button.</p> <p><i>Usage note:</i></p> <p>Click Find 200 before Find All to prevent unwanted matches in case the Highlight Search (*) string is less selective than you expect.</p>
Find All	<p>Places all objects that match the Highlight Search (*) filter string in the UnHighlighted list. This button does not find objects in HDL Analyst views. It matches names of design objects against the Highlight Search (*) filter and provides the candidates listed in the UnHighlighted list, from which you select the objects to find. (Enter a filter string before clicking this button.) See <i>Usage Note</i> for Find 200, above.</p>

For more information on using the Object Query dialog box, see [Using Find for Hierarchical and Restricted Searches, on page 279](#) of the *User Guide*.

Find in Files Command

The Find in Files command searches the defined target for the occurrence of a specified search string. The list of files containing the string is reported in the display area at the bottom of the dialog box. For information on using this feature, see [Searching Files, on page 96](#) of the *User Guide*.

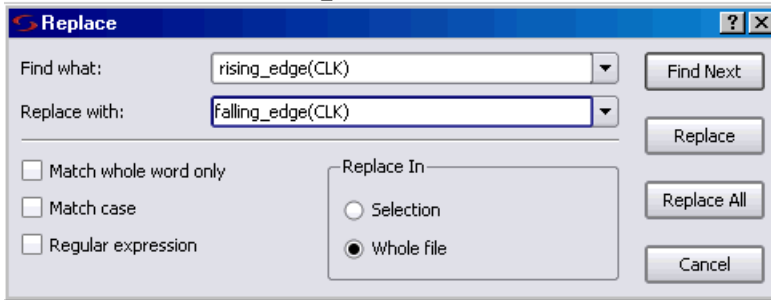


Field/Option	Description
Find what	Text string object of search.
Files Contained in Project	Drop-down menu identifying the source project of the files to be searched.
Implementation Directory	Drop-down menu restricting project search to a specific implementation or all implementations.
Directory	Identifies directory for files to be searched.

Field/Option	Description
Result Window	Allows a secondary search string (Find what) to be applied to the targets reported from the initial search.
Include SubProject files	When checked, extends the search to include subproject files of the top-level project file.
Include sub-folders for directory searches	When checked, extends the search to sub-directories of the target directory.
File filter	Excludes files from the search by filename extension.
Search Options	Standard string search options; check to enable.
Find	Initiates search.
Result Display	List of files containing search string. Status line lists the number of matches in each file and the number of files searched.

Replace Command

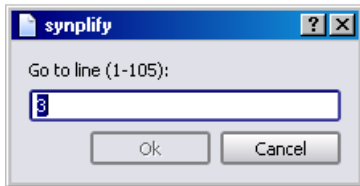
Use Edit->Replace to find and optionally replace text in the Text Editor.



Feature	Description
Find what	Search string matching the text to find. You can use the pull-down list to view and reuse search strings used previously in the current session.
Replace with	The text that replaces the found text. You can use the pull-down list to view and reuse replacement text used previously in the current session.
Match whole word only	Finds only occurrences of the exact string (strings longer than the Find what string are not recognized).
Match case	When enabled, searching is case sensitive.
Regular expression	When enabled, wildcard characters (* and ?) can be used in the search string: ? matches any single character; * matches any string of characters, including the empty string.
Selection	Replace All replaces only the matched occurrence.
Whole file	Replace All replaces all matching occurrences.
Find Next	Initiates a search for the search string (see Find What).
Replace	Replaces the found text with the replacement text, and locates the next match.
Replace All	Replaces all text that matches the search string.

Goto Command

Use Edit->Goto to go to a specified line number in the Text Editor.



View Menu

Use the View menu to set the display and viewing options, choose toolbars, and display result files. The commands in the View menu vary with the active view. The following tables describe the View menu commands in various views.






- [View Menu Commands: All Views, on page 191](#)
- [View Menu: Zoom Commands, on page 192](#)
- [View Menu: RTL and Technology Views Commands, on page 192](#)
- [View Menu: FSM Viewer Commands, on page 193](#)

View Menu Commands: All Views

Command	Description
Font Size	Changes the font size in the Project UI of the synthesis tools. You can select one of the following options: <ul style="list-style-type: none"> • Increase Font Size • Decrease Font Size • Reset Font Size (default size)
Toolbars	Displays the Toolbars dialog box, where you specify the toolbars to display. See Toolbar Command, on page 194 .
Status Bar	When enabled, displays context-sensitive information in the lower-left corner of the main window as you move the mouse pointer over design elements. This information includes element identification.
Refresh	Updates the UI display of project files and folders.
Output Windows	Displays or removes the Tcl Script/Messages and Watch windows simultaneously in the Project view. Refer to the Tcl Window and Watch Window options for more information.
Tcl Window	When enabled, displays the Tcl Script and Messages windows. All commands you execute in the Project view appear in the Tcl window. You can enter or paste Tcl commands and scripts in the Tcl window. Check for notes, warning, and errors in the Messages window.






Command	Description
Watch Window	When enabled, displays selected information from the log file in the Watch window.
View Log File	Displays a log file report that includes compiler, mapper, and timing information on your design. See View Log File Command, on page 196 .
View Result File	Displays a detailed netlist report.

View Menu: Zoom Commands

Command	Description
 Zoom In	Lets you Zoom in or out. When selected, a Z-shaped mouse pointer () appears. Zoom in or out on the view by clicking or dragging a box around (lassoing) the region. Clicking zooms in or out on the center of the view; lassoing zooms in or out on the lassoed region. Right-click to exit zooming mode. In the SCOPE spreadsheet, selecting these commands increases or decreases the view in small increments.
 Zoom Out	
Pan	Lets you pan (scroll) a schematic or FSM view using the mouse. If your mouse has a wheel feature, use the wheel to pan up and down. To pan left and right, use the Shift key with the wheel.
 Full View	Zooms the active view so that it shows the entire design.
 Normal View	Zooms the active view to normal size and centers it where you click. If the view is already normal size, clicking centers the view.



View Menu: RTL and Technology Views Commands



These commands are available when the RTL view or Technology view is active. These commands are available in addition to the commands described in [View Menu Commands: All Views, on page 191](#) and [View Menu: Zoom Commands, on page 192](#).

Command	Description
 Push/Pop Hierarchy	Traverses design hierarchy using the push/pop mode – see Exploring Design Hierarchy, on page 269 of the <i>User Guide</i> .
 Previous Sheet	Displays the previous sheet of a multiple-sheet schematic.
 Next Sheet	Displays the next sheet of a multiple-sheet schematic.
View Sheets	Displays the Goto Sheet dialog box where you can select a sheet to display from a list of all sheets. See View Sheets Command, on page 195 .
Visual Properties	<p>Toggles the display of information for nets, instances, pins, and ports in the HDL Analyst view.</p> <p>To customize the information that displays, set the values with Options->HDL Analyst Options->Visual Properties. See Visual Properties Panel, on page 338.</p>
 Back	Goes backward in the history of displayed sheets for the current HDL Analyst view.
 Forward	Goes forward in the history of displayed sheets for the current HDL Analyst view.
Filter	Filters the RTL/Technology view to display only the selected objects.

View Menu: FSM Viewer Commands

The following commands are available when the FSM viewer is active. These commands are in addition to the common commands described in [View Menu Commands: All Views, on page 191](#) and [View Menu: Zoom Commands, on page 192](#).

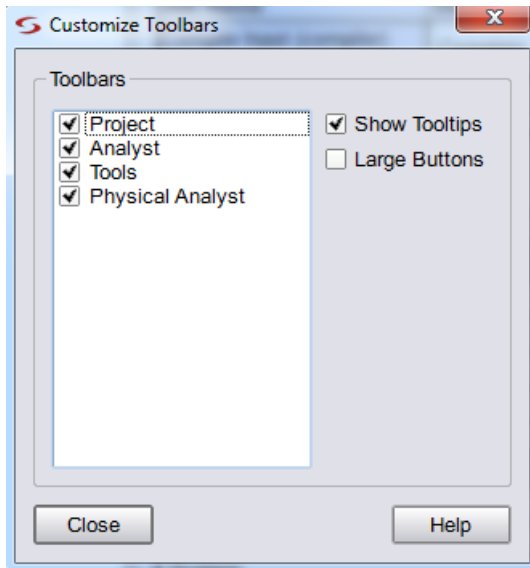
Command	Description
 Filter->Selected	Hides all but the selected state(s).
 Filter->By output transitions	Hides all but the selected state(s), their output transitions, and the destination states of those transitions.
Filter->By input transitions	Hides all but the selected state(s), their input transitions, and the origin states of those transitions.

Command	Description
Filter->By any transition	Hides all but the selected state(s), their input and output transitions, and their predecessor and successor states.
 Unfilter	Restores a filtered FSM diagram so that all the states and transitions are showing.
Cross Probing	Enables cross probing between FSM nodes and RTL view schematic.
Select All States	Selects all the states.
 FSM Table	Toggles display of the transition table.
FSM Graph	Toggles FSM state diagram on or off.
Annotate Transitions	Toggles display of state transitions on or off on FSM state diagram.
Selection Transcription	
Tool Tips	Toggles state diagram tool tips on or off.
FSM Properties	Displays FSM Properties dialog box.
Unselect All	Unselects all states and transitions.

Toolbar Command

Select View->Toolbars to display the Toolbars dialog box, where you can:

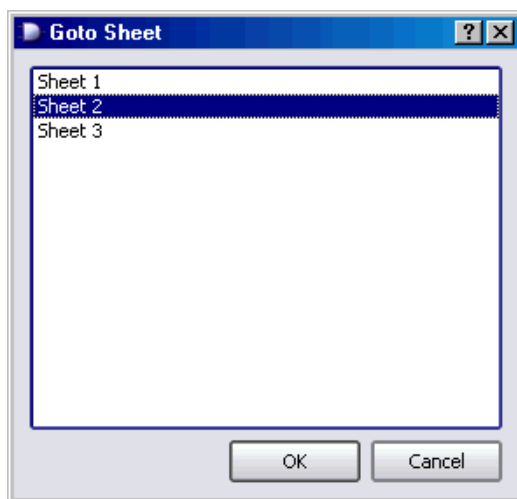
- Choose the toolbars to display
- Customize their appearance



Feature	Description
Toolbars	Lists the available toolbars. Select the toolbars that you want to display.
Show Tooltips	When selected, a descriptive tooltip appears whenever you position the pointer over an icon.
Large Buttons	When selected, large icons are used.

View Sheets Command

Select View->View Sheets to display the Goto Sheet dialog box and select a sheet to display. The Goto Sheet dialog box is only available in an RTL or Technology view, and only when a multiple-sheet design is present.

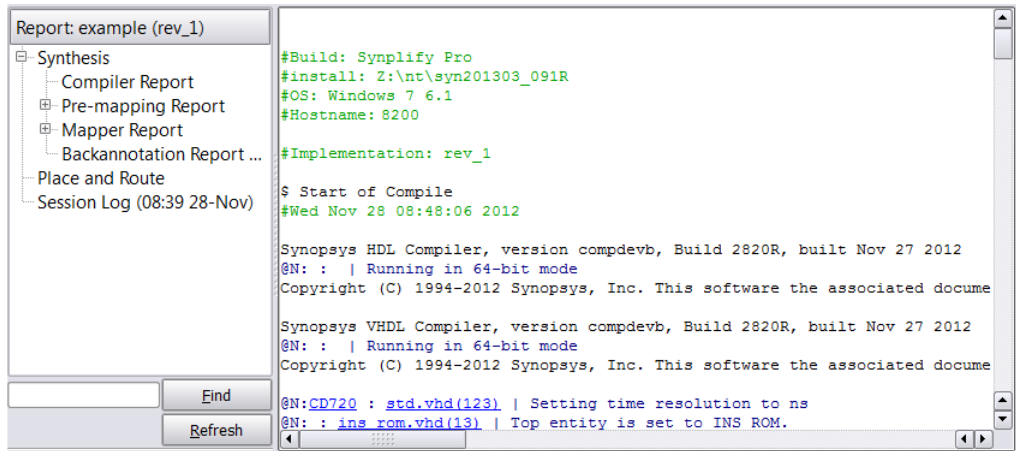


To see if your design has multiple sheets, check the sheet count display at the top of the schematic window.

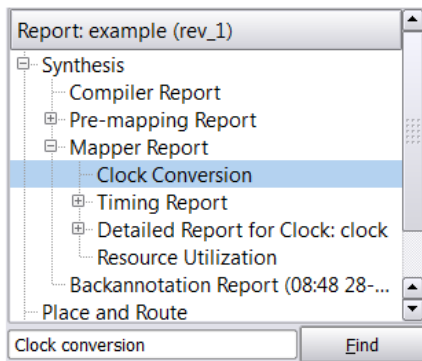
View Log File Command

View->View Log File displays the log file report for your project. The log file is available in either text (*project_name.srr*) or HTML (*project_name_srr.htm*) format. To enable or disable the HTML file format for the log file, select the View log file in HTML option in the Options->Project View Options dialog box.

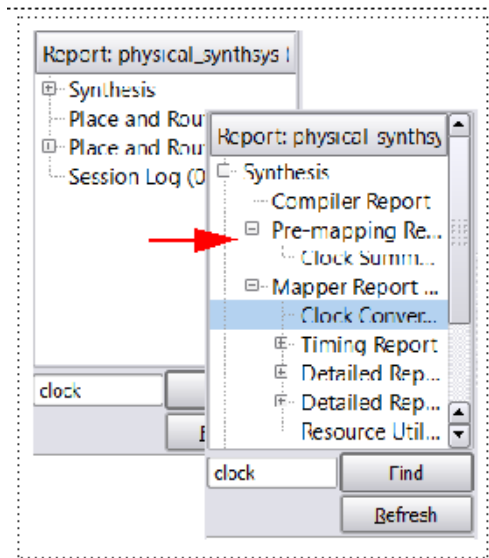
When opening the log file, a table of contents appears. Selecting an item from the table of contents takes you to the corresponding HTML page. To go back, right-click on the HTML page and select Back from the menu.



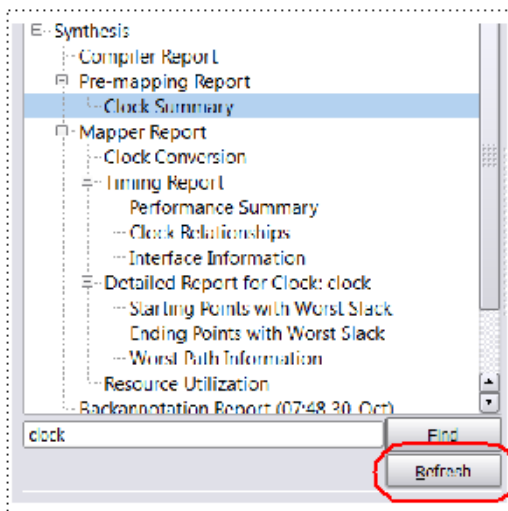
You can use the search field to find an item in the table of contents. Enter all or part of the header name in the search field, then click Find. The log file displays the resulting section.



Find searches within collapsed tables. It expands the tables to show your results.



If the file changes while the search window is open, click the Refresh button to update the table of contents.



Project Menu

You use the Project menu to set implementation options, add or remove files from a project, change project filenames, create new implementations, and archive or copy the project. The Project menu commands change, depending on the view you are in. For example, the HDL Analyst RTL and Technology views only include a subset of the Project menu commands.

The tool provides a graphical user interface (GUI) with views that help you manage hierarchical designs that can be synthesized independently and imported back to the top-level project in a team design flow called Hierarchical Project Management. This feature is not available for Microsemi technologies.

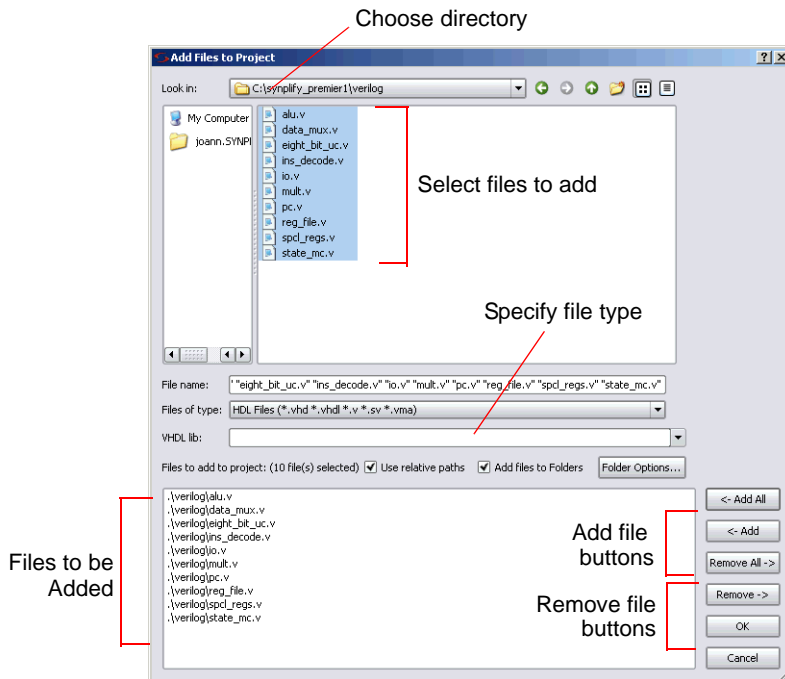
The following table describes the Project menu commands.

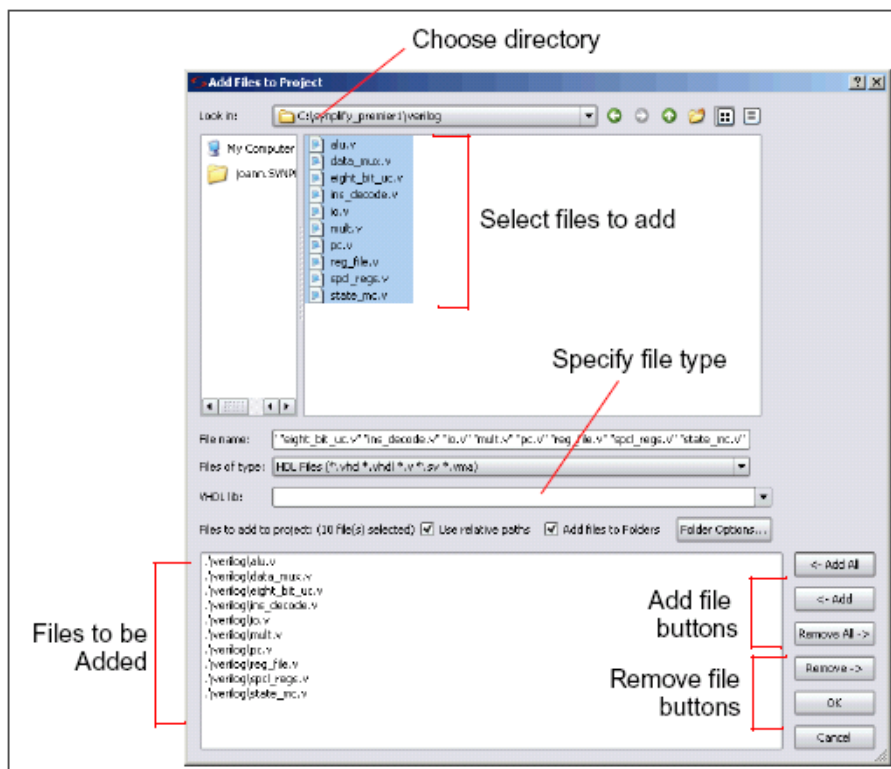
Command	Description
Implementation Options	Displays the Implementation Options dialog box, where you set options for implementing your design. See Implementation Options Command, on page 214 .
Add Source File	Displays the Select Files to Add to Project dialog box. See Add Source File Command, on page 200 . Tcl equivalent: add_file -fileType <i>filename</i>
Remove Implementation	Displays the Remove Implementation dialog box that allows you to remove the selected implementation. See Remove Implementation, on page 203 . Tcl equivalent: impl -remove <i>implementationName</i>
Remove File From Project	Removes selected files from your project. Tcl equivalent: project_file -remove <i>filename</i>
Change File	Replaces the selected file in your project with another that you choose. See Change File Command, on page 204 . Tcl equivalent: project_file -name " <i>originalFile</i> " " <i>newFile</i> "
Set VHDL Library	Displays the File Options dialog box, where you choose the library (Library Name) for synthesizing VHDL files. The default library is called work. See Set VHDL Library Command, on page 204 .

Command	Description
Add Implementation	<p>Creates a new implementation for a current design. Each implementation pertains to the same design, but it can have different options settings and/or constraints for synthesis runs. See Add Implementation Command, on page 205).</p> <p>Tcl equivalent: impl -add <i>implementation_1 implementation -type implementationType</i></p>
New Identify Implementation	<p>Creates a new Identify implementation for a current design. To launch the Identify toolset, see the Identify Instrumentor Command, on page 250 and Launch Identify Debugger Command, on page 252.</p> <p>Tcl equivalent: impl -add <i>implementation_1 implementation</i></p>
Convert Vendor Constraints	Not applicable for Microsemi technologies.
Archive Project	<p>Archives a design project. Use this command to archive a full or partial project, or to add files to or remove files from an archived project. See Archive Project Command, on page 206 for a description of the utility wizard options.</p>
Un-Archive Project	<p>Loads an archived project file to the specified directory. See Un-Archive Project Command, on page 207 for a description of the utility wizard options.</p>
Copy Project	<p>Creates a copy of a full or partial design project. See Copy Project Command, on page 210 for a description of the utility wizard options.</p>
Hierarchical Project Options	Not applicable for Microsemi technologies.
Add SubProject Implementation	Not applicable for Microsemi technologies.

Add Source File Command

Select Project->Add Source File to add files, such as HDL source files, to your project. This selection displays the Select Files to Add to Project dialog box.





Feature	Description
Look in	The directory of the file to add. You can use the pull-down directory list or the Up One Level button to choose the directory.
File name	The name of a file to add to the project. If you enter a name using the keyboard, then you must include the file-type extension.
Files of type	The type (extension) of files to be added to the project. Only files in the active directory that match the file type selected from the drop-down menu are displayed in the list of files. Use All Files to list all files in the directory.

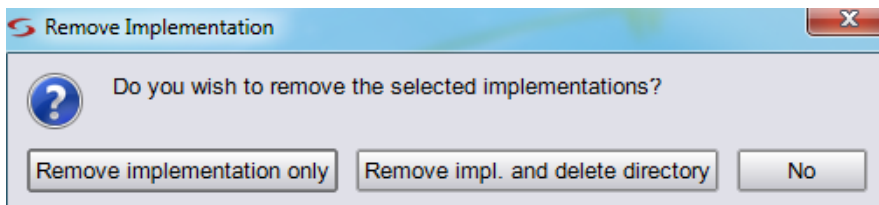
Feature	Description
Files To Add To Project	<p>The files to add to the project. You add files to this list with the <-Add and <-Add All buttons. You remove files from this list with the Remove -> and Remove All -> buttons.</p> <p>For information about adding files to custom folders, see Creating Custom Folders, on page 64.</p> <p>Tcl equivalent: add_file -type filename</p>
Use relative paths	When you add files to the project, you can specify either to use the relative path or full path names for the files.
Add files to Folders	When you add files to the project, you can specify whether or not to automatically add the files to folders. See the Folder Options described below.
Folder Options	<p>When you add files to folders, you can specify the folder name as either the:</p> <ul style="list-style-type: none"> • Operating System (OS) folder name • Parent path name from a list provided in the display

Remove Implementation

Displays the Remove Implementation dialog box that allows you to remove the selected implementation. You can select any of the following:

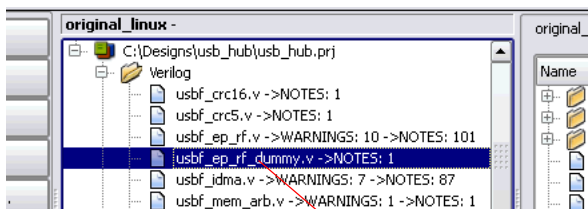
- Remove implementation only – Removes the implementation from the project only.
- Remove impl. and delete directory – Removes the implementation from the project and deletes the directory on the disk.
- No – Does not remove the implementation.

Choose the appropriate option shown in the dialog box below.



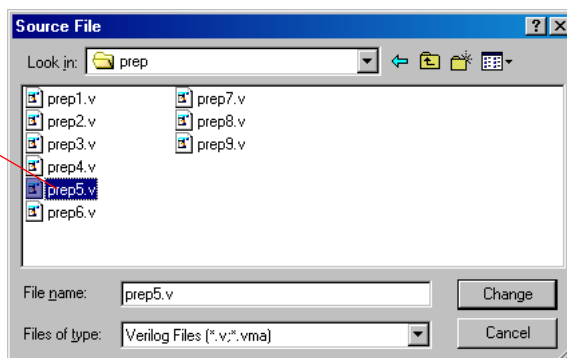
Change File Command

Select Project->Change File to replace a file in the project files list with another of the same type. This displays the Source File dialog box, where you specify the replacement file. You must first select the file to replace, in the Project view, before you can use this command.



First select a file in the Project view

Then choose the replacement file



Set VHDL Library Command

Select Project->Set VHDL Library to display the File Options dialog box, where you view VHDL file properties and specify the VHDL library name. See [File Options Popup Menu Command, on page 359](#). This is the same dialog box as that displayed by right-clicking a VHDL filename in the Project view and choosing File Options.

Add Implementation Command

Select Project->Add Implementation to create a new implementation for the selected project. This selection displays the Implementation Options dialog box, where you define the implementation options for the project – see [Implementation Options Command, on page 214](#). This is the same dialog box as that displayed by Project->Implementation Options, except that there is no list of Implementations to the right of the tabbed panels.

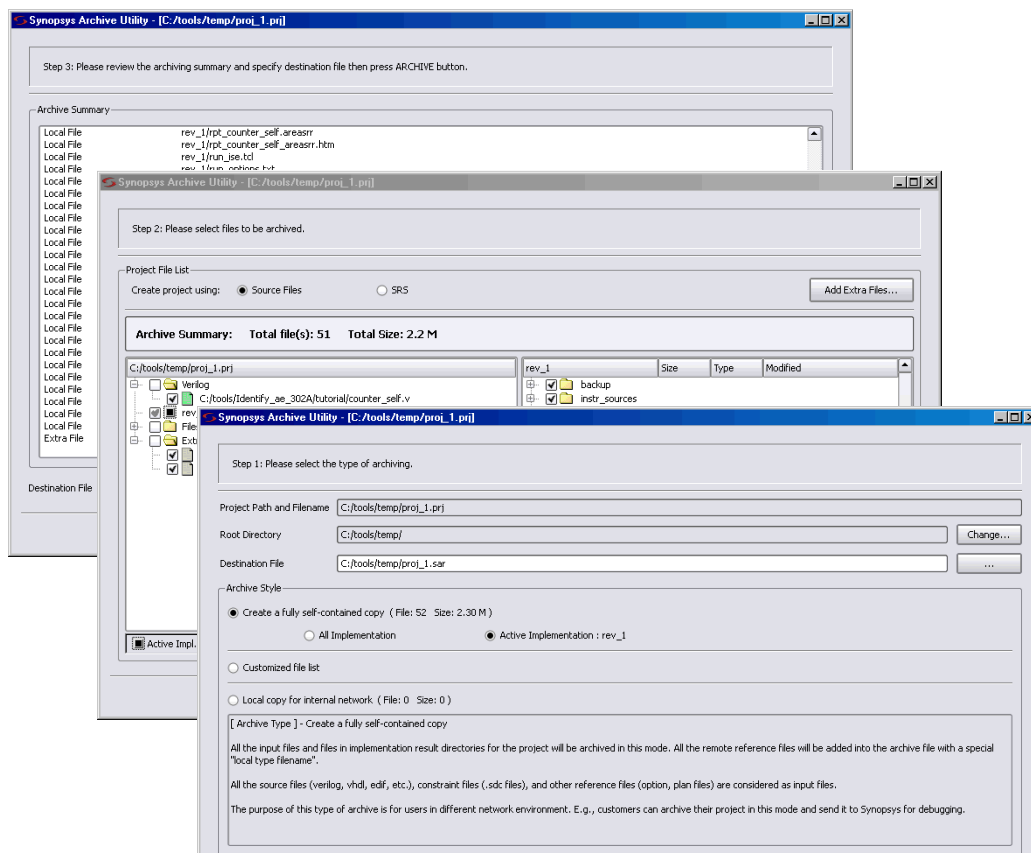
Convert Vendor Constraints Command

The Project->Convert Vendor Constraints is not available for Microsemi technologies.

Archive Project Command

Use the Project->Archive Project command to store files for a design project into a single archive file in Synopsys Proprietary Format (sar). You can archive an entire project or selected files from the project.

The Archive Project command displays the Synopsys Archive Utility wizard consisting of either two (all files archived) or three (custom file selection) tabs.



Option	Description
Project Path and Filename	Path and filename of the .prj file.
Root Directory	Top-level directory that contains the project files.
Destination Directory	Pathname of the directory to store the archive .sar file.

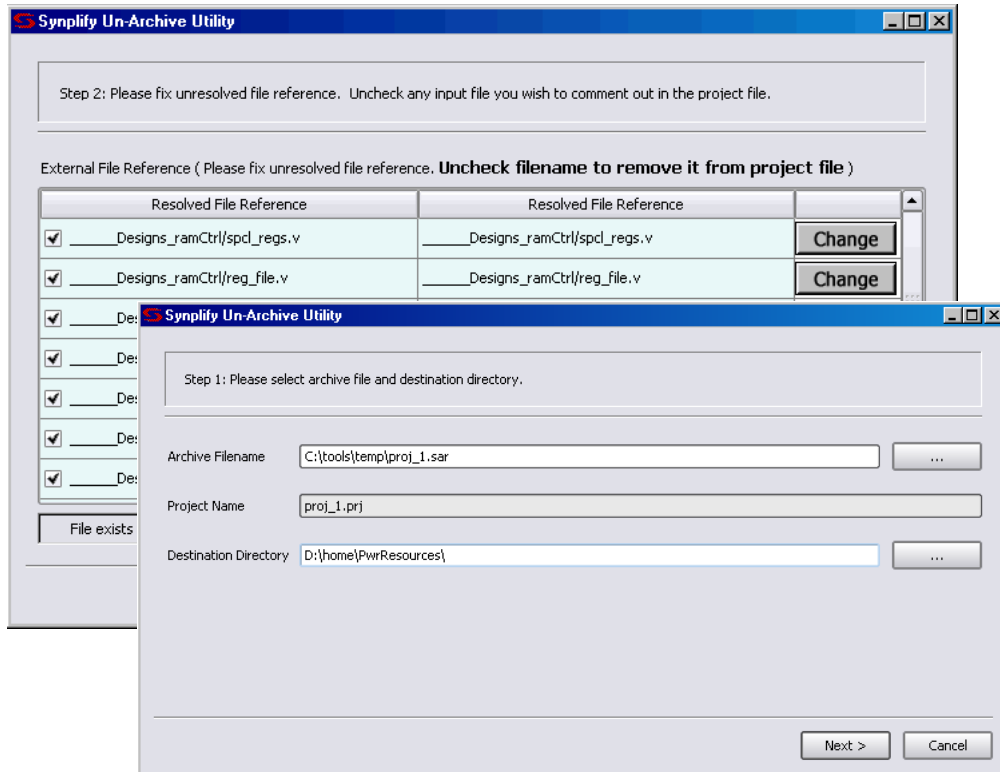
Option	Description
Archive Style	<p>The type of archive:</p> <ul style="list-style-type: none"> • Create a fully self-contained copy – all project files are archived; includes project input files and result files. • If the project contains more than one implementation: <ul style="list-style-type: none"> - All Implementation includes all implementations in the project. - Active Implementation includes only the active implementation. • Customized file list – only project files that you select are included in the archive. • Local copy for internal network – only project input files are archived, no result files will be included.
Create Project using	<p>If you select the Customized file list option in the wizard, you can choose one the following options on the second tab:</p> <ul style="list-style-type: none"> • Source Files – Includes all design files in the archive. You cannot enable the SRS option if this option is enabled. • SRS – Includes all .srs files (RTL schematics) in the archive. You cannot enable the Source Files option when this option is enabled.
Add Extra Files	<p>If you select the Customized file list option in the wizard, you can use this button on the second tab to add additional files to the archive.</p>

For step-by-step details on how to use the archive utility, see [Archive Project Command, on page 206](#).

Un-Archive Project Command

Use the Project->Un-Archive Project command to extract the files from an archived design project.

This command displays a Synplify Un-Archive Utility wizard.

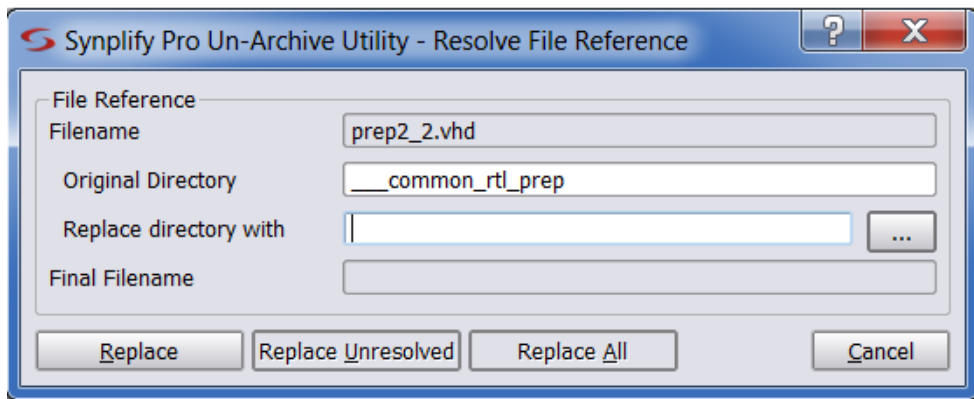


Option	Description
Archive Filename	Path and filename of the .sar file.
Project Name	Top-level directory that contains the project files.
Destination Directory	Pathname of the directory to store the archive .sar file.
Original File Reference/ Resolved File Reference	<p>Displays the files in the archive that will be extracted. You can exclude files from the .sar by unchecking the file in the Original File Reference list. Any unchecked files are commented out in the .prj file.</p> <p>If there are unresolved reference files in the .sar file, you must fix (Resolve button) or uncheck them. Or, if there are files that you want to change when project files are extracted, use the Change button and select files, as appropriate. See Resolve File Reference, next for more details.</p>

For step-by-step details on how to use the un-archive utility, see [Un-Archive Project Command](#), on page 207.

Resolve File Reference

When you use the Un-Archive Utility wizard to extract a project, if there are unresolved file references, use the Resolve button next to the file to point to a new file location. You can also optionally replace project files in the destination directory by clicking the Change button next to the file you want to replace. The Change and Resolve buttons bring up the following dialog box:



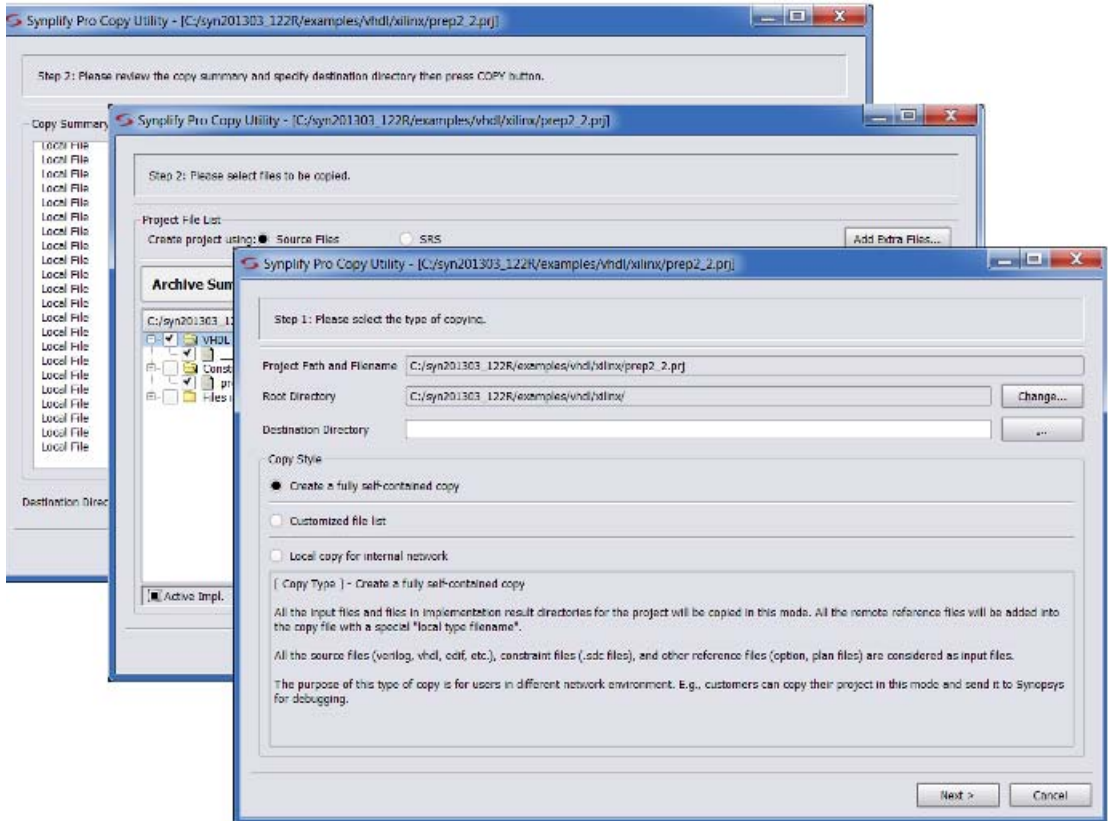
Option	Description
Filename	Specifies the path and name of the file you want to change or resolve.
Original Directory	Specifies the location of the project at the time it was archived.

Option	Description
Replace directory with	Specifies the new location of the project files you want to use to replace files.
Final Filename	Specifies the path name of the directory and the file name of the replace file.
Replace buttons	<ul style="list-style-type: none">• Replace – replaces only the file specified in the Filename field when the project is extracted.• Replace Unresolved – replaces any unresolved files in the project, with files of the same name from the Replace directory.• Replace All – replaces all files in the archived project with files of the same name from the Replace directory.• To undo any replace-file references, clear the Replace directory with field, then click Replace. This causes the utility to point back to the Original Directory and filenames.

Copy Project Command

Use the Project->Copy Project command to create a copy of a design project. You can copy an entire project or selected files from the project.

The Copy Project command displays the Synopsys Copy Utility wizard consisting of either two (all files copied) or three (custom file selection) tabs.



Option

Description

Project Path and Filename	Path and filename of the .prj file.
Root Directory	Top-level directory that contains the project files.
Destination Directory	Pathname of the directory to store the archive .sar file.

Option	Description
Copy Style	<p>The type of archive:</p> <ul style="list-style-type: none">• Create a fully self-contained copy – all project files are archived; includes project input files and result files.• If the project contains more than one implementation:<ul style="list-style-type: none">- All Implementation includes all implementations in the project.- Active Implementation includes only the active implementation.• Customized file list – only project files that you select are included in the archive.• Local copy for internal network – only project input files are archived, no result files will be included.
Create Project using	<p>If you select the Customized file list option in the wizard, you can choose one the following options on the second tab:</p> <ul style="list-style-type: none">• Source Files – Includes all design files in the archive. You cannot enable the SRS option if this option is enabled.• SRS – Includes all .srs files (RTL schematics) in the archive. You cannot enable the Source Files option if this option is enabled.
Add Extra Files	<p>If you select the Customized file list option in the wizard, you can use this button on the second tab to add additional files to the archive.</p>

For step-by-step details on how to use the copy utility, see [Copy Project Command, on page 210](#).

Hierarchical Project Options Command

The Project->Hierarchical Project Options command is not available for Microsemi technologies.

Implementation Options Command

You use the Implementation Options dialog box to define the implementation options for the current project. You can access this dialog box from Project->Implementation Options, by clicking the button in the Project view, or by clicking the text in the Project view that lists the current technology options.

The dialog box displays the following configuration:

- Technology:** Microsemi PolarFire
- Part:** PA5M300
- Package:** FBGA896
- Speed:** STD

Device Mapping Options

Option	Value
Fanout Guide	10000
Disable I/O Insertion	<input type="checkbox"/>
Update Compile Point Timing Data	<input type="checkbox"/>
Annotated Properties for Analyst	<input checked="" type="checkbox"/>
Operating Conditions	COMTC
Max Number of Critical Paths in SDF	4000
Conservative Register Optimization	<input type="checkbox"/>
Automatic Read/Write Check Insertion for RAM	<input type="checkbox"/>
Resolve Mixed Drivers	<input type="checkbox"/>

Click on an option for description

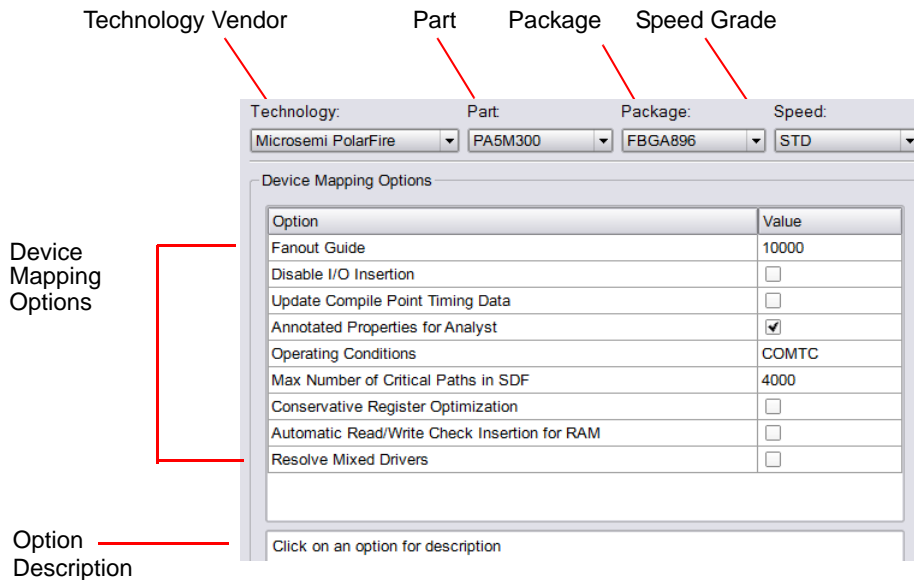
This section describes the following:

- [Device Panel, on page 215](#). For device-specific details of the options, refer to the appropriate vendor chapter.
- [Options Panel, on page 217](#)
- [Constraints Panel, on page 218](#)
- [Implementation Results Panel, on page 220](#)
- [Timing Report Panel, on page 222](#)
- [High Reliability Panel, on page 223](#)

- [VHDL Panel, on page 228](#)
- [Compiler Directives and Design Parameters, on page 231](#)
- [Place and Route Panel, on page 242](#)

Device Panel

You use the Device panel to set mapping options for the selected technology.



The mapping options vary, depending on the technology. See [Setting Device Options, on page 72](#) in the *User Guide* for a procedure, and the relevant vendor sections in this reference manual for technology-specific descriptions of the options.

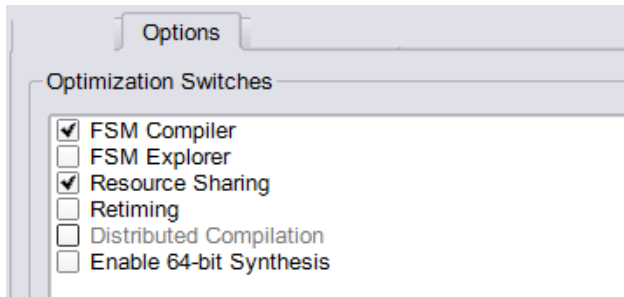
The table below lists the following category of options. Not all options are available for all tools and technologies.

Option	Description
Technology Vendor	Specify the device technology you want to synthesize. You can also select the part, package, and speed grade to use. For more information, see the appropriate vendor appendix in the <i>Reference</i> manual.
Device Mapping Options	The device mapping options vary depending on the device technology you select. For more information, see the appropriate vendor appendix in the <i>Reference</i> manual.
Option Description	Click a device mapping option to display its description in this field. Refer to the relevant vendor sections for technology-specific descriptions of the options.

Options Panel

You use the Options panel of the Implementation Options dialog box to define general options for synthesis optimization. See [Setting Optimization Options, on page 75](#) of the *User Guide* for details.

The following table lists the options alphabetically. Not all options are avail-



able for all tools and technologies.

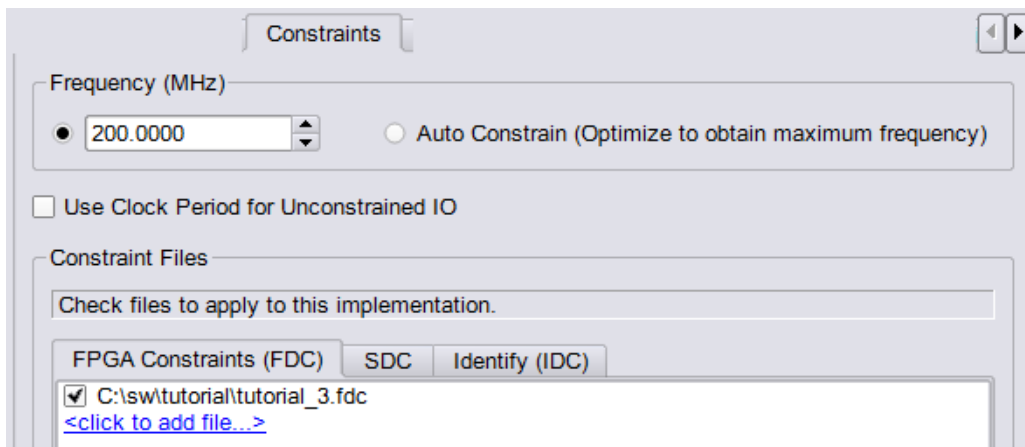
Option	Description
FSM Compiler	Determines whether the FSM Compiler is run. See FSM Compiler, on page 67 and Optimizing State Machines, on page 404 in the <i>User Guide</i> . Tcl equivalent: set_option -symbolic_fsm_compiler 0 1
FSM Explorer	Determines whether the FSM Explorer is run. See Running the FSM Explorer, on page 409 in the <i>User Guide</i> . Tcl equivalent: set_option -use_fsm_explorer 0 1

Option	Description
Resource Sharing	<p>Determines whether you optimize area by sharing resources. When enabled, this optimization technique runs during the compilation stage of synthesis.</p> <p>Even if it is disabled, the mapper can still flatten the netlist and re-optimize adders, multipliers as needed to improve timing, because this setting does not affect the mapper. See Sharing Resources, on page 402 for information for how to use this option in the <i>User Guide</i>.</p> <p>Enabling this option generates the resource sharing report in the log file (see Resource Usage Report, on page 258).</p> <p>Tcl equivalent: set_option -resource_sharing 0 1</p>
Retiming	<p>Determines whether the tool moves storage devices across computational elements to improve timing performance in sequential circuits. Note that the tool might retime registers associated with RAMs, DSPs, and generated clocks, regardless of the Retiming setting.</p> <p>See Retiming, on page 386 in the <i>User Guide</i>.</p> <p>Tcl equivalent: set_option -retiming 0 1</p>
Enable 64-bit Synthesis	<p>Enables/disables the 64-bit mapping switch. When enabled, this switch allows you to run client programs in 64-bit mode, if available on your system.</p> <p>This option is supported on the Windows and Linux platforms.</p> <p>Tcl equivalent: set_option -enable64bit 0 1</p>

Constraints Panel

You use the Constraints panel of the Implementation Options dialog box to specify target frequency and timing constraint files for design synthesis. Depending on the synthesis tool you are using and the device you specify, the types of constraint files you can apply for the implementation may vary. See the table below for a complete list of option types you can apply.

See [Specifying Global Frequency and Constraint Files, on page 77](#), in the *User Guide* for details.



Option	Description
Frequency	<p>Sets the default global frequency. You can either set the global frequency here or in the Project view. To override the default you set here, set individual clock constraints from the SCOPE interface.</p> <p>Tcl equivalent: set_option -frequency frequency</p>
Auto Constrain	<p>When enabled and no clocks are defined, the software automatically constrains the design to achieve the best possible timing. It does this by reducing periods of each individual clock and the timing of any timed I/O paths in successive steps. See Using Auto Constraints, on page 342 in the <i>User Guide</i> for information about using this option.</p> <p>You can also set this option in the Project view.</p> <p>Tcl equivalent: set_option -frequency auto</p>

Option	Description
Use clock period for unconstrained IO	<p>Determines whether default constraints are used for I/O ports that do not have user-defined constraints.</p> <p>When disabled, only <code>set_input_delay</code> or <code>set_output_delay</code> constraints are considered during synthesis or forward-annotated after synthesis.</p> <p>When enabled, the software considers any explicit <code>set_input_delay</code> or <code>set_output_delay</code> constraints. In addition, for all ports without explicit constraints, it uses constraints based on the clock period of the attached registers. Both the explicit and implicit constraints are used for synthesis and forward-annotation. The default is off (disabled) for new designs.</p> <p>Tcl equivalent: <code>set_option -auto_constrain_io 0 1</code></p>
Constraint Files SDC	<p>Specifies which timing constraints (SDC) files to use for the implementation. Select the check box to select a file.</p> <p>For the block-level files in the compile-point flows, the Module column shows the name of the module or compile point.</p>
Constraint Files FDC	<p>Specifies which timing constraints (FDC) files to use for the implementation. Select the check box to select a file.</p> <p>For the block-level files in the compile-point flows, the Module column shows the name of the module or compile point.</p>
Identify (IDC)	<p>Specifies the instrumentation design constraints (IDC) files that add compiler pragmas in these files to the design RTL for the instrumented signals and break points. Enable the checkbox to select a file.</p>

Implementation Results Panel

You use the Implementation Results panel to specify the implementation name (default: `rev_1`), the results directory, and the name and format of the top-level output netlist file (Result File). You can also specify output constraint and netlist files. See [Specifying Result Options, on page 79](#) of the *User Guide* for details.

The results directory is a subdirectory of the project file directory. Clicking the Browse button brings up the Select Run Directory dialog box to allow you to browse for the results directory. You can change the location of the results directory, but its name must be identical to the implementation name.

Select optional output file check boxes to generate the corresponding Verilog netlist, VHDL netlist, or vendor constraint files.

The screenshot shows the 'Implementation Results' dialog box. On the left, there are four labels with red lines pointing to specific fields in the dialog:

- Implementation name** points to the 'Implementation Name' field, which contains 'rev_1'.
- Results directory** points to the 'Results Directory' field, which contains 'C:\synplify_pro_actel\rev_1'. A 'Browse...' button is to the right.
- Result filename** points to the 'Results File Name' field, which contains 'eight_bit_uc.edf'. A 'Result Format' dropdown menu is to the right, currently set to 'edif.n'.
- Result format** points to the 'Result Format' dropdown menu.
- Optional output files** points to a section titled 'Optional Output Files' which contains three checkboxes:
 - ☐ Write Mapped Verilog Netlist
 - ☐ Write Mapped VHDL Netlist
 - ☒ Write Vendor Constraint File

Option	Description
Implementation Name	Displays implementation name, directory path for results, and the base name for the result files.
Results Directory	Tcl equivalent: set_option -result_file <i>pathtoResultFile</i>
Result Base Name	

Option	Description
Result Format	Select the output that corresponds to the technology you are using. See Generating Vendor-Specific Output, on page 548 in the <i>User Guide</i> for a list of netlist formats. Tcl equivalent: set_option -result_format <i>format</i>
Write Mapped Verilog Netlist	Generates mapped Verilog or VHDL netlist files. Tcl equivalent: set_option -write_verilog 0 1
Write Mapped VHDL Netlist	Tcl equivalent: set_option -write_vhdl 0 1
Write Vendor Constraint File	Generates a vendor-specific constraint file for forward annotation. Tcl equivalent: set_option -write_apr_constrain 0 1

Timing Report Panel

Use the Timing Report panel (Implementation Options dialog box) to set criteria for the (default) output timing report. Specify the number of critical paths and the number of start and end points to appear in the timing report. See [Specifying Timing Report Output, on page 81](#) in the *User Guide* for details. For a description of the report, see [Timing Reports, on page 259](#).

Timing Report

Number of Critical Paths:

Number of Start/End Points:

Description

Configure the timing report by specifying the number of paths to include in the "Starting/Ending Points with worst slack" and "Worst Paths" report sections.

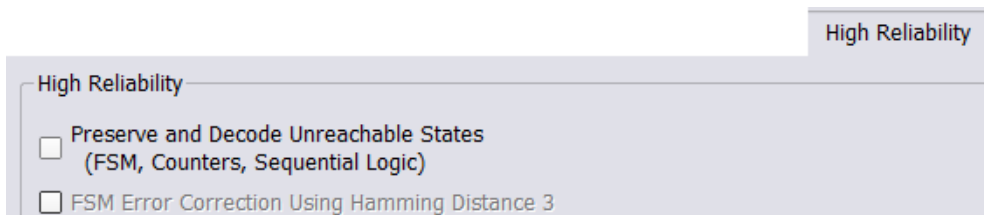
Option	Description
Number of Critical Paths	Set the number of critical paths for the software to report. Tcl equivalent: set_option -num_critical_paths <i>numberOfPaths</i>
Number of Start/End Points	Specify the number of start and end points to see reported in the critical path sections. Tcl equivalent: set_option -num_startend_points <i>numberOfPoints</i>

See also:

- [Timing Reports, on page 259](#), for more information on the default timing report, which is affected by the Timing Report panel settings.
- [Analysis Menu, on page 291](#), information on creating additional custom timing reports for certain device technologies.

High Reliability Panel

Use the High Reliability panel (Implementation Options dialog box) to implement safe logic for the design.



Option	Description
Preserve and Decode Unreachable States (FSM, Counters, Sequential Logic)	When enabled, this option turns off sequential optimizations on all counters, FSMs, and sequential logic, to increase the reliability of the circuit. If you do not want to implement this globally, use the <code>syn_safe_case</code> directive (syn_safe_case, on page 189) on individual FSMs. Tcl equivalent: set_option -safe_case 0 1

Verilog Panel

You use the Verilog panel in the Implementation Options dialog box to specify various language-related options. With mixed HDL designs, the VHDL and Verilog panels are both available. See [Setting Verilog and VHDL Options, on page 81](#) of the *User Guide* for details.

Feature	Description
Top Level Module	The name of the top-level module of your design. Tcl equivalent: set_option -top_module <i>moduleName</i>
Compiler Directives and Parameters	Shows design parameters extracted with Extract Parameters. You can override the default and set a new value for the parameter. The value is valid for the current implementation.
Extract Parameters	Extracts design parameters from the top-level module and displays them in the table. See Compiler Directives and Design Parameters, on page 231 .

Feature	Description
Compiler Directives	Provides an interface where you can enter compiler directives that you would normally enter in the code with 'ifdef and 'define statements. See Compiler Directives and Design Parameters , on page 231.
Verilog Language – Verilog 2001	<p>When enabled, the default Verilog standard for the project is Verilog 2001. When both Verilog 2001 and SystemVerilog are disabled, the default standard is Verilog 95. For information about Verilog 2001, see Verilog 2001 Support, on page 27.</p> <p>You can override the default project standard on a per file basis by selecting the file, right-clicking, and selecting the File Options command (see File Options Popup Menu Command, on page 359).</p> <p>Tcl equivalent: set_option -vlog_std v2001</p>
Verilog Language – SystemVerilog	<p>When enabled, the default Verilog standard for the project is SystemVerilog which is the default standard for all new projects. Enabling SystemVerilog automatically enables Verilog 2001.</p> <p>Tcl equivalent: set_option -vlog_std sysv</p>
Push Tristates	<p>When enabled (default), tristates are pushed across process/block boundaries. For details, see Push Tristates Option, on page 240.</p> <p>Tcl equivalent: set_option -compiler_compatible 0 1</p>
Allow Duplicate Modules	<p>Allows the use of duplicate modules in your design. When enabled, the last definition of the module is used by the software and any previous definitions are ignored.</p> <p>You should not use duplicate module names in your Verilog design, therefore this option is disabled by default. However, if you need to, you can allow for duplicate modules by enabling this option.</p> <p>Tcl equivalent: set_option -dup 0 1</p>
Multiple File Compilation Unit	<p>When enabled (the default), the Verilog compiler uses the compilation unit for modules defined in multiple files. See SystemVerilog Compilation Units, on page 155 for additional information.</p> <p>Tcl equivalent: set_option -multi_file_compilation_unit 0 1</p>

Feature	Description
Beta Features for Verilog	<p>Enables use of any Verilog beta features included in the release. Enabling this checkbox is equivalent to including a <code>set_option -hdl_define -set _BETA_FEATURES_ON_</code> directive in the project file.</p> <p>Tcl equivalent: <code>set_option -beta_vfeatures 0 1</code></p>
Auto Infer Blackbox	<p>Selects how undefined modules are treated in the synthesis tool. Selecting None (the default) causes the compiler to error out when an undefined module is encountered. Selecting With BIDIR ports creates a black box for the undefined module and generates a warning message, but allows the operation to continue; this mode matches the behavior of Certify.</p> <p>Tcl equivalent: <code>set_option -auto_infer_blackbox 0 1</code></p>
Loop Limit	<p>Allows you to override the default compiler loop limit value of 2000 in the RTL. You can apply loop limits using the Verilog <code>loop_limit</code> or the VHDL <code>syn_looplmit</code> directive.</p> <p>For details about these directives, see loop_limit, on page 37 and syn_looplmit, on page 113 in the <i>Attribute Reference</i>.</p> <p>Tcl equivalent: <code>set_option -looplmit loopLimitValue</code></p>
Include Path Order (Relative to Project File)	<p>Specifies the search paths for the include commands in the Verilog design files of your project. Use the buttons in the upper right corner of the box to add, delete, or reorder the paths. The include paths are relative. See Updating Verilog Include Paths in Older Project Files, on page 63 in the <i>User Guide</i> for additional information.</p>

Feature	Description
Library Directories or Files	<p>Specifies all the paths to the directories which contain the Verilog library files to be included in your design for the project. You can also add custom library files with module definitions for the design in a single file. See Verilog Single Library File Support, on page 241. The names of files read from the library path must match module names. Mismatches result in error messages.</p> <p>Tcl equivalent:</p> <p>set_option -library_path <i>./libraryPath</i> or <i>libraryFile</i></p>
Library Extensions (space separated)	<p>Adds library extensions to Verilog library files included in your design for the project and searches the directory paths you specified that contain these Verilog library files. To use library extensions, see Using Library Extensions for Verilog Library Files, on page 40 in the <i>User Guide</i>.</p> <p>Tcl equivalent:</p> <p>set_option -libext <i>.libextName1 .libextName1...</i></p> <p>Enter a space between each unique library extension.</p>

VHDL Panel

You use the VHDL panel in the Implementation Options dialog box to specify various language-related options. With mixed HDL designs, the VHDL and Verilog panels are both available. See [Setting Verilog and VHDL Options, on page 81](#), of the *User Guide* for details.

The screenshot shows the VHDL panel of the Implementation Options dialog box. The panel has a tab labeled "VHDL". It contains the following elements:

- Top Level Entity:** A text field containing "eight_bit_uc".
- Default Enum Encoding:** A dropdown menu set to "default".
- Options:** A list of checkboxes:
 - ☒ Push Tristates
 - ☐ Synthesis On/Off Implemented as Translate On/Off
 - ☐ VHDL 2008
 - ☐ Implicit Initial Value Support
 - ☐ Beta Features for VHDL
- Loop Limit:** A text field containing "2000".
- Generics:** A section with a table for defining generic constants.
- Extract Generic Constants:** A button at the bottom right.

Generic Name	Override Value

The following table describes the options available.

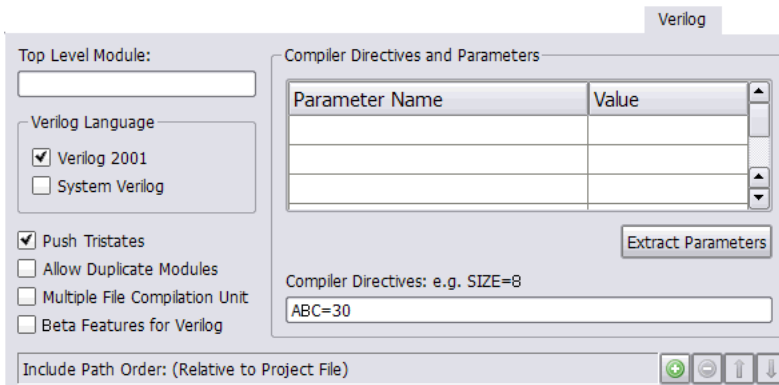
Feature	Description
Top Level Entity	<p>The name of the top-level entity of your design.</p> <p>If the top-level entity does not use the default work library to compile the VHDL files, you must specify the library file where the top-level entity can be found. To do this, the top-level entity name must be preceded by the VHDL library followed by a period (.). To specify VHDL library files, see Project Menu, on page 199 for the Set VHDL Library command, or the File Options Popup Menu Command, on page 359.</p> <p>Tcl equivalent: set_option -top_module <i>topLevelName</i></p>
Default Enum Encoding	<p>The default enumeration encoding to use. This is only for enumerated types; the FSM compiler automatically determines the state-machine encoding, or you can specify the encoding using the <code>syn_encoding</code> attribute.</p> <p>Tcl equivalent: set_option -default_enum_encoding <i>encodingType</i></p>
Push Tristates	<p>When enabled (default), tristates are pushed across process/block boundaries. For more information, see Push Tristates Option, on page 240.</p> <p>Tcl equivalent: set_option -compiler_compatible 0 1</p>
Synthesis On/Off Implemented as Translate On/Off	<p>When enabled, the software ignores any VHDL code between <code>synthesis_on</code> and <code>synthesis_off</code> directives. It treats these third-party directives like <code>translate_on/translate_off</code> directives (see translate_off/translate_on, on page 229 for details).</p> <p>Tcl equivalent: set_option -synthesis_onoff_pragma 0 1</p>
VHDL 2008	<p>When enabled, allows you to use VHDL 2008 language standards.</p> <p>Tcl equivalent: set_option -vhdl2008 0 1</p>
Implicit Initial Value Support	<p>When enabled, the compiler passes init values through a <code>syn_init</code> property to the mapper. For more information, see VHDL Implicit Data-type Defaults, on page 233.</p> <p>Tcl equivalent: set_option -supporttypedflt 0 1</p>

Feature	Description
Beta Features for VHDL	<p>Enables use of any VHDL beta features included in the release. Enabling this checkbox is equivalent to including a <code>set_option -hdl_define -set _BETA_FEATURES_ON_</code> directive in the project file.</p> <p>Tcl equivalent: <code>set_option -beta_vhfeatures 0 1</code></p>
Loop Limit	<p>Allows you to override the default compiler loop limit value of 2000 in the RTL. You can apply loop limits using the Verilog <code>loop_limit</code> or the VHDL <code>syn_looplmit</code> directive.</p> <p>For details about these directives, see loop_limit, on page 37 and syn_looplmit, on page 113 in the <i>Attribute Reference</i>.</p> <p>Tcl equivalent: <code>set_option -looplimit loopLimitValue</code></p>
Generics	<p>Shows generics extracted with Extract Generic Constants. You can override the default and set a new value for the generic constant. The value is valid for the current implementation.</p>
Extract Generic Constants	<p>Extracts generics from the top-level entity and displays them in the table.</p>

Compiler Directives and Design Parameters

When you click the Extract Parameters button in the Verilog panel (Implementation Options dialog box), parameter values from the top-level module are displayed in the table. You can also override the default by setting a new value for the parameter. The value is valid for the current implementation only.

The Compiler Directives field provides an interface where you can enter compiler directives that you would normally enter in the code using 'ifdef and 'define statements. Use spaces to separate the statements. The directives you enter are stored in the project file. For example, if you enter the directive shown below to the Compiler Directives field of the Verilog panel:



The software writes the following statement to the project file:

```
set_option -hdl_define -set "ABC=30"
```

To define multiple variables in the GUI, use a space delimiter. For example:

The screenshot shows the 'Verilog' tab of a configuration window. On the left, under 'Verilog Language', 'Verilog 2001' is checked. Below it, 'Push Tristates' is checked, while 'Allow Duplicate Modules', 'Multiple File Compilation Unit', and 'Beta Features for Verilog' are unchecked. On the right, the 'Compiler Directives and Parameters' section contains a table with two columns: 'Parameter Name' and 'Value'. Below the table, the text 'Compiler Directives: e.g. SIZE=8' is followed by a text box containing 'ABC=30 XYZ=12 vj'. An 'Extract Parameters' button is to the right of this text box. At the bottom, there is a section for 'Include Path Order: (Relative to Project File)' with four navigation icons.

The software writes the following statement to the prj file:

```
set_option hdl_define -set "ABC=30 XYZ=12 vj"
```

More information is provided for the following Verilog compiler directives:

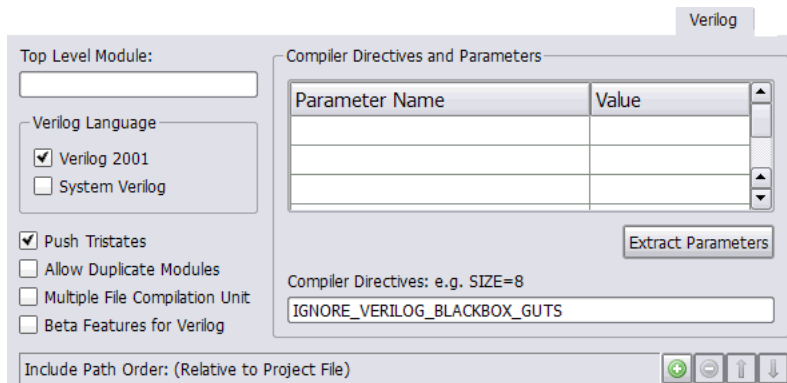
Directive	Description
<u>__ALLOWNESTEDBLOCKCOMMENTSTART__</u>	Allows for nested comment blocks.
<u>__BETA_FEATURES_ON__</u>	Explicitly enables beta HDL language features.
<u>IGNORE_VERILOG_BLACKBOX_GUTS</u>	Ignores the contents of a black box.
<u>__SEARCHFILENAMEONLY__</u>	Provides workaround for archive utility limitations.
<u>SYN_COMPATIBLE</u>	Ensures compatibility of Synopsys tools such as Design Compiler (DC) with the synthesis software.
<u>__SYN_COMPATIBLE_INCLUDEPATH__</u>	Specifies that the search path order for includes to be the same as the one used by the simulation tool (VCS).

IGNORE_VERILOG_BLACKBOX_GUTS

When you use the `syn_black_box` directive, the compiler parses the contents of the black box and can determine whether illegal syntax or incorrect code is defined within it. Whenever this occurs, an error message is generated. You can specify the `IGNORE_VERILOG_BLACKBOX_GUTS` compiler directive to ignore the contents of the black box. However, make sure that the black box is syntactically correct.

If you want the tool to ignore the contents of your black box, set the:

- Built-in compiler directive `IGNORE_VERILOG_BLACKBOX_GUTS` in the Compiler Directives field of the Verilog panel on the Implementation Options dialog box.



The software writes the following command to the project file:

```
set_option -hdl_define -set "IGNORE_VERILOG_BLACKBOX_GUTS"
```

- ``define IGNORE_VERILOG_BLACKBOX_GUTS` directive in the Verilog file.

This option is implemented globally for the project file.

Example of the IGNORE_VERILOG_BLACKBOX_GUTS Directive

Note that the `IGNORE_VERILOG_BLACKBOX_GUTS` directive ignores the contents of the black box. However, whenever you use this directive, you must first define the ports for the black box correctly. Otherwise, the `IGNORE_VERILOG_BLACKBOX_GUTS` directive generates an error. See the following valid Verilog example:

```

`define IGNORE_VERILOG_BLACKBOX_GUTS
module b1_fpga1 (A,B,C,D) /* synthesis syn_black_box */;
input B;
output A;
input [2:0] D;
output [2:0] C;
temp;
assign A = B;
assign C = D;

endmodule

module b1_fpga1_top (inout A, B, inout [2:0] C, D);
b1_fpga1 b1_fpga1_inst(A,B,C,D);
endmodule

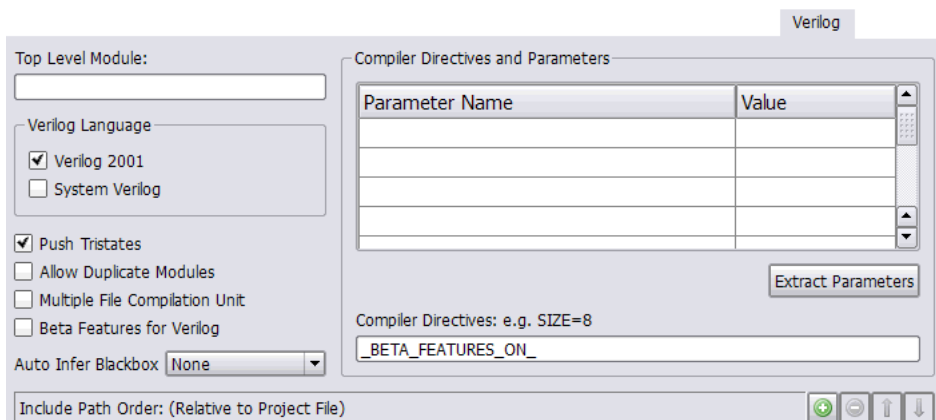
```

_BETA_FEATURES_ON_

Beta features for the Verilog, SystemVerilog, or VHDL language must be explicitly enabled. In the UI, a Beta Features checkbox is included on the VHDL or Verilog tab of the Implementations Options dialog box. A `_BETA_FEATURES_ON_` compiler directive is also available. This directive is specified with a `set_option -hdl_define` command added to the project file as shown below:

```
set_option -hdl_define -set _BETA_FEATURES_ON_
```

The directive can also be added to the Compiler Directives field of the Verilog panel.



Current beta features that must be explicitly enabled for the compiler include the following:

HDL Language Constructs	Descriptions
-------------------------	--------------

Aggregates	<ul style="list-style-type: none"> • Multi-assignments for array aggregates • Assignments for the union of variable types
------------	---

__SEARCHFILENAMEONLY__

This directive provides a workaround for some known limitations of the archive utility.

If Verilog 'include files belong in any of the following categories, you may encounter problems when compiling a design after un-archiving:

1. The include paths have relative paths to the project file.

```
`include "../../../defines.h"
```

2. The include paths have absolute paths to the project file.

```
`include "c:/temp/params.h"
```

```
`include "/remote/sbg_home/user/params.h"
```

3. The include paths have the same file names, but are located in different directories relative to the project file.

```
`include "../myflop.v"
```

```
...
```

```
`include "../../myflop.v"
```

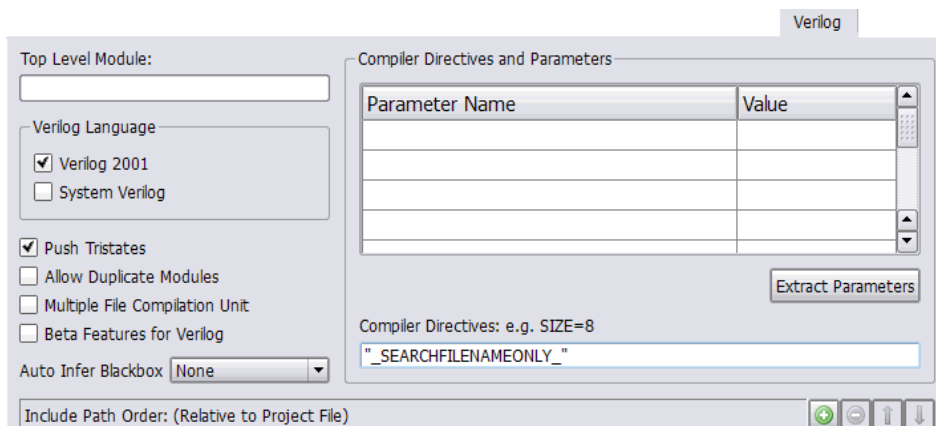
Use the `__SEARCHFILENAMEONLY__` directive to resolve categories 1 and 2 above. Category 3 is a known limitation; in this case it is recommended that you adopt standard coding practices to avoid files with the same name and different content.

When you un-archive a sar file that contains relative or absolute include paths for the files in the project, you can add the `__SEARCHFILENAMEONLY__` compiler directive to the unarchived project. This has the compiler remove the relative/absolute paths from the ``include` and search only for the file names.

This directive is specified with a `set_option -hdl_define` command added to an implementation within the project file as shown below:

```
set_option -hdl_define -set "__SEARCHFILENAMEONLY__"
```

The directive can also be added to the Compiler Directives field of the Verilog panel as shown below.



The compiler generates the following warning message whenever it extracts include files using this directive:

```
@W: | Macro _SEARCHFILENAMEONLY_ is set: fileName not found
attempting to search for base file name fileName
```

__ALLOWNESTEDBLOCKCOMMENTSTART__

Verilog/SystemVerilog comments can be included in RTL code as a

- One-line comment starting with the characters `//` and ending with a new line
- Block comment starting with `/*` and ending with `*/`

However, nested block comments are not supported according to the LRM, so most tools generate a warning and ignore these comments in the RTL code. To match this behavior, the synthesis tools must also ignore various nested block comments, such as:

```
/* ...../* ....*/
```

(An incorrect pair)

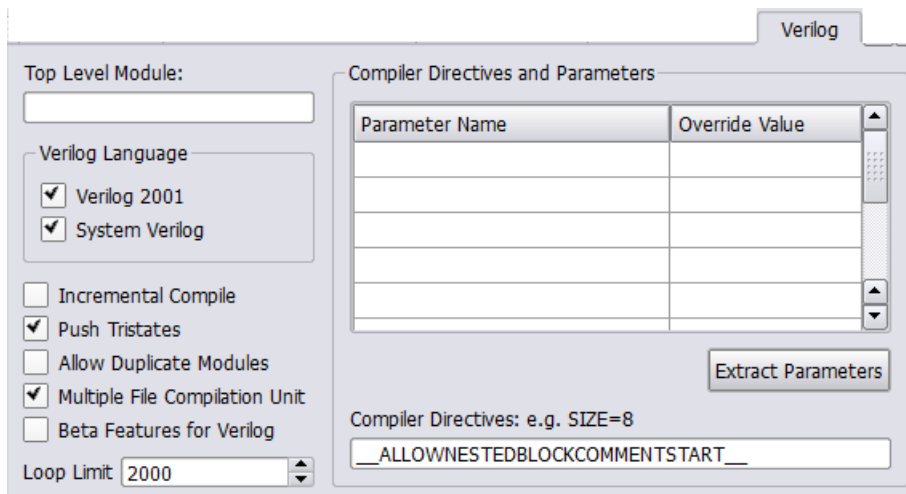

```
/*...../*...../*.....*/
```

To do this, the compiler uses the `__ALLOWNESTEDBLOCKCOMMENTSTART__` directive and parses the first `/*` until it encounters the associated pair `*/`, then ignores all content between and including any number of these lines and the block comments.

You can specify the `__ALLOWNESTEDBLOCKCOMMENTSTART__` directive with a `set_option -hdl_define -set` command added to the project file as shown below:

```
set_option -hdl_define -set __ALLOWNESTEDBLOCKCOMMENTSTART__
```

The directive can also be added to the Compiler Directives field of the Verilog panel.



SYN_COMPATIBLE

Use the `SYN_COMPATIBLE` macro to ensure compatibility between different Synopsys tools. Some Synopsys tools, such as Design Compiler (DC), ignore dynamic initialization assignments, unlike the synthesis tool. In the following example, note the line `logic a=b;` DC leaves the output unconnected, because DC does not handle inline assignments. By contrast, the synthesis tool handles inline assignments, so input `b` drives the output `q`.

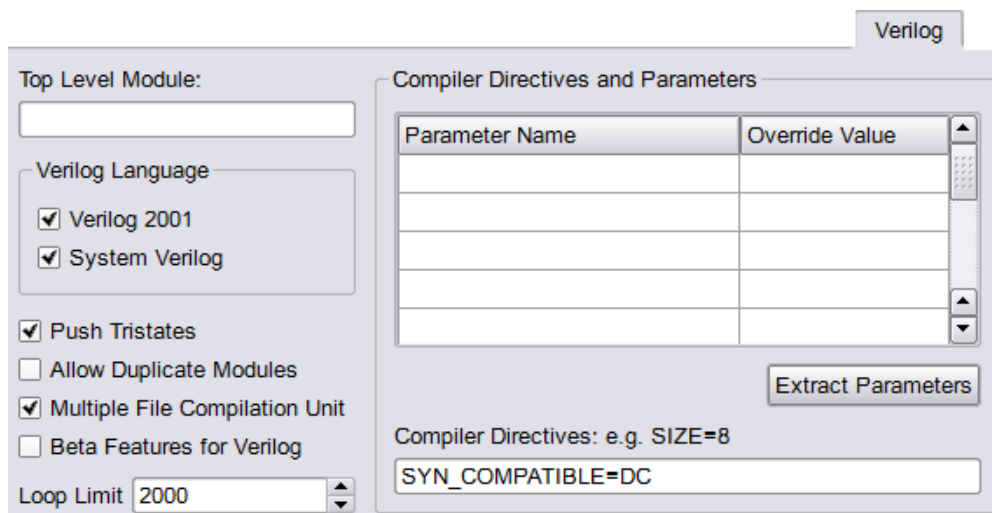
```

module test (b, q);
input b;
output q;
logic a = b;
assign q = a;
endmodule

```

The tool warns you of the difference in handling (warning message @W: CG879), and treats the assignment as a regular assign. If you are using modules from DC and need it to be compatible with the synthesis tool, there are two ways to match the behavior and ignore the non-constant initial values:

1. Specify a macro with a Tcl command.
 - set_option -hdl_define -set SYN_COMPATIBLE=DC
2. Specify a macro through the GUI.
 - To specify it from the GUI, go to the Verilog tab of the Implementation Options dialog box.
 - In the Compiler Directives field, set SYN_COMPATIBLE=DC.



__SYN_COMPATIBLE_INCLUDEPATH__

Specifies that the search path order for includes to be the same as the one used by the simulation tool (VCS), instead of the following default search order, which searches the current logical library of the file where the module is instantiated first, then the library path for the current logical library, and lastly other logical libraries.

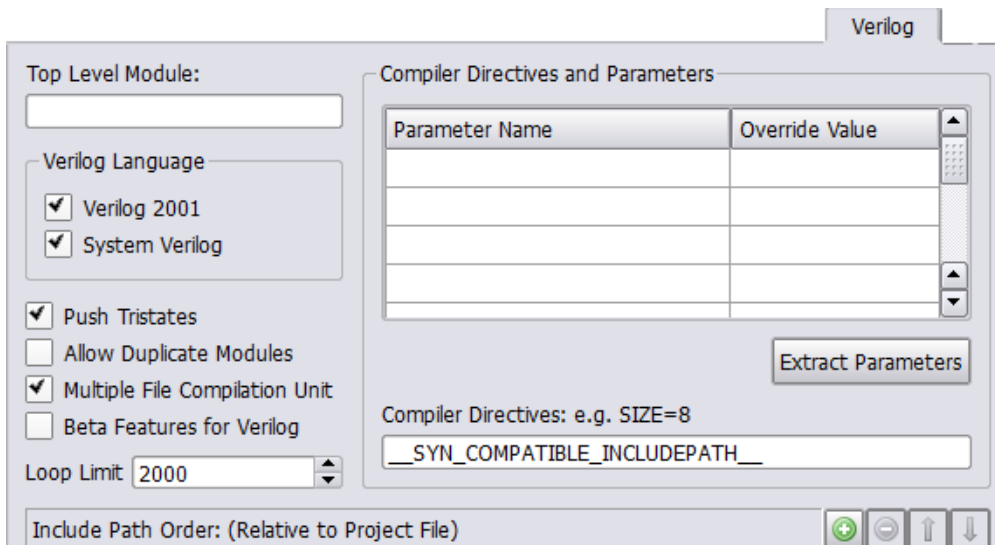
With __SYN_COMPATIBLE_INCLUDEPATH__, this is the search path order:

- Current working directory
- Include file directories, with first priority to RTL includes and then include paths

You can specify the __SYN_COMPATIBLE_INCLUDEPATH__ directive with a `set_option -hdl_define` command added to the project file as shown below:

```
set_option -hdl_define -set __SYN_COMPATIBLE_INCLUDEPATH__
```

The directive can also be added to the Compiler Directives field of the Verilog panel.

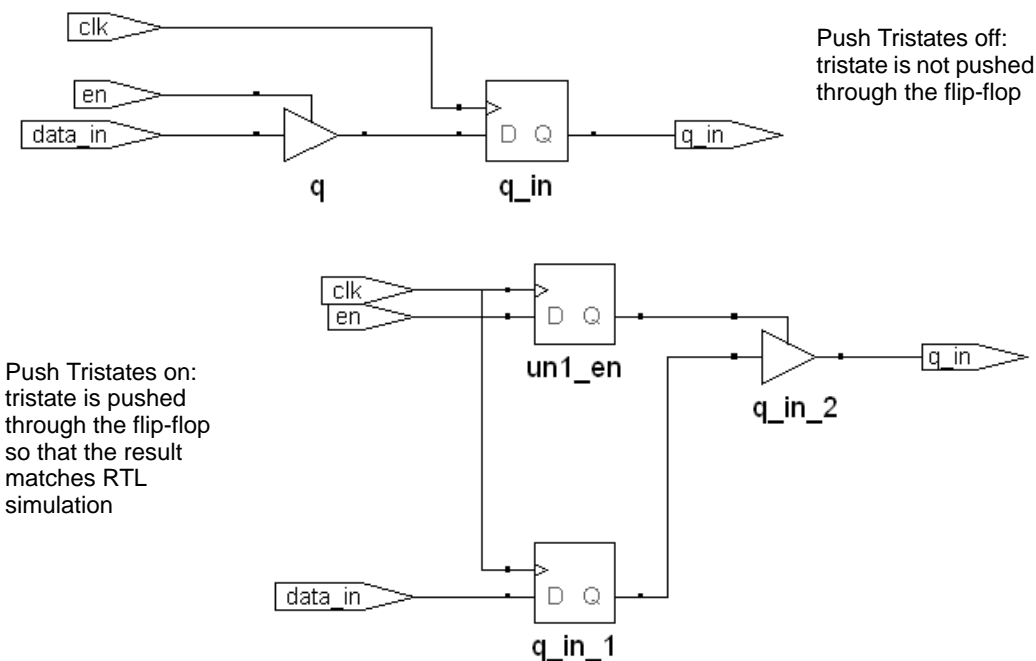


Push Tristates Option

Pushing tristates is a synthesis optimization option you set with Project->Implementation Options->Verilog or VHDL.

Description

When the Push Tristates option is enabled, the Synopsys FPGA compiler pushes tristates through objects such as muxes, registers, latches, buffers, nets, and tristate buffers, and propagates the high-impedance state. The high-impedance states are not pushed through combinational gates such as ANDs or ORs.



If there are multiple tristates, the software muxes them into one tristate and pushes it through. The software pushes tristates through loops and stores the high impedance across multiple cycles in the register.

Advantages and Disadvantages

The advantage to pushing tristates to the periphery of the design is that you get better timing results because the software uses tristate output buffers.

The Synopsys FPGA software approach to tristate inference matches the simulation approach. Simulation languages are defined to store and propagate 0, 1, and Z (high impedance) states. Like the simulation tools, the Synopsys FPGA synthesis tool propagates the high-impedance states instead of producing tristate drivers at the outputs of process (VHDL) or always (Verilog) blocks.

The disadvantage to pushing tristates is that you might use more design resources.

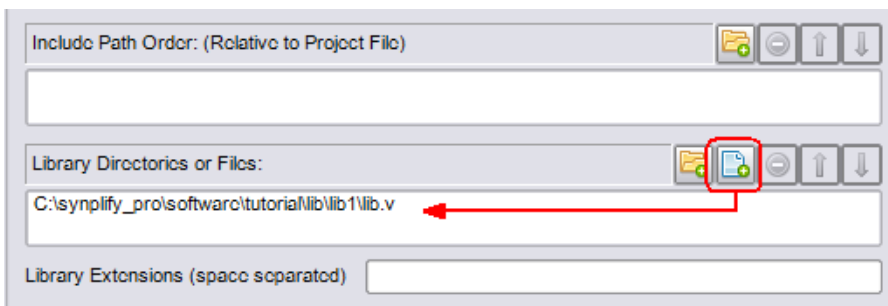
Effect on Other Synthesis Options

Tristate pushing has no effect on the `syn_tristatetomux` attribute. This is because tristate pushing is a compiler directive, while the `syn_tristatetomux` attribute is used during mapping.

Verilog Single Library File Support

You can add a single library file to your project for easier migration from a VCS environment and to ensure their behaviors are consistent for the design. To do this, either:

- Select the Add a file icon from the Verilog tab of the Implementation Options panel. Then, specify the library file to be added to the project from the Library Directories and Files option.

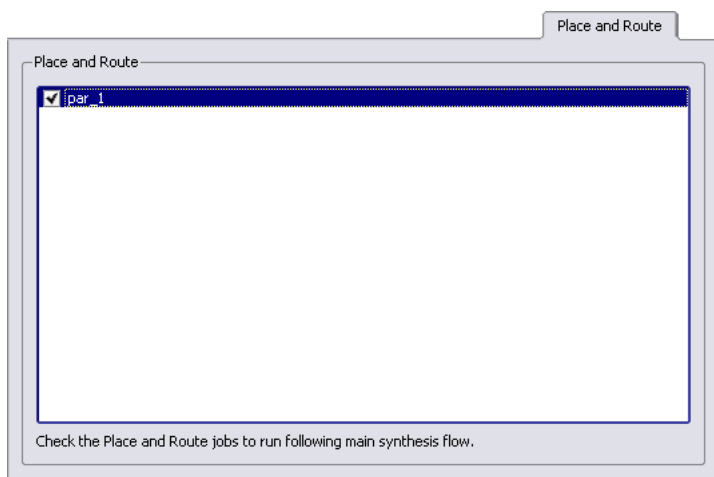


- Add the following Tcl command to your project file:

```
set_option -library_path {./libPath/libFile.v}
```

Place and Route Panel

The Place and Route panel allows you to run selected place-and-route jobs after design synthesis. To create a place-and-route job, see [Add P&R Implementation Popup Menu Command, on page 363](#) or [Options for Place & Route Jobs Popup Menu Command, on page 366](#) for details.



Import Menu

The Import option is not available for Microsemi technologies.

Run Menu

You use the Run menu to perform the following tasks:

- Compile a design, without mapping it.
- Synthesize (compile and map) or resynthesize a design.
- Check design syntax and synthesis code, and check source code errors.
- Check constraint syntax and how/if constraints are applied to the design.
- Run Tcl scripts.
- Run all implementations at once.
- Check the status of the current job.

The following table describes the Run menu commands.

Command	Description
Run	<p>Synthesizes (compiles and maps) the top-level design. For the compile point flow, this command also synthesizes any compile points whose constraints, implementation options, or source code changed since the last synthesis run. You can view the result of design synthesis in the RTL and Technology views.</p> <p>Same as clicking the Run button in the Project view.</p> <p>Tcl equivalent: project -run</p>
Resynthesize All	<p>Resynthesizes (compiles and maps) the entire design, including the top level and <i>all compile points</i>, whether or not their constraints, implementation options, or source code changed since the last synthesis. If you do <i>not</i> want to force a <i>recompilation of all compile points</i>, then use Run->Run instead.</p> <p>Tcl equivalent: project -run synthesis -clean</p>
Compile Only	<p>Compiles the design into technology-independent high-level structures. You can view the result in the RTL view.</p> <p>Tcl equivalent: project -run compile</p>

Command	Description
Write Output Netlist Only	<p>Generates an output netlist after synthesis has been run. This command generates the netlists you specify on the Implementation Results tab of the Implementation Options dialog box.</p> <p>You can also use this command in an incremental timing analysis flow. See Generating Custom Timing Reports with STA, on page 332 for details.</p> <p>Tcl equivalent: project -run write_netlist</p>
FSM Explorer	<p>Analyzes finite state machines contained in a design, and selects the optimum encoding style. This menu command is not available in some views.</p> <p>Tcl equivalent: project -run fsm_explorer</p>
Syntax Check	<p>Runs a syntax check on design code. The status bar at the bottom of the window displays any error messages. If the active window shows an HDL file, then the command checks only that file; otherwise, it checks all project source code files.</p> <p>Tcl equivalent: project -run syntax_check</p>
Synthesis Check	<p>Runs a synthesis check on your design code. This includes a syntax check and a check to see if the synthesis tool could map the design to the hardware. No optimizations are carried out. The status bar at the bottom of the window displays any error messages. If the active window shows an HDL file, then the command checks only that file; otherwise, it checks all project source code files.</p> <p>Tcl equivalent: project -run synthesis_check</p>
Constraint Check	<p>Checks the syntax and applicability of the timing constraints in the fdc/sdc file for your project and generates a report (<i>projectName_cck.rpt</i>). The report contains information on the constraints that can be applied, cannot be applied because objects do not exist, and wildcard expansion on the constraints.</p> <p>See Constraint Checking Report, on page 268.</p> <p>Tcl equivalent: project -run constraint_check</p>
Arrange VHDL files	<p>Reorders the VHDL source files for synthesis.</p> <p>Tcl equivalent: project -run hdl_info_gen fileorder</p>

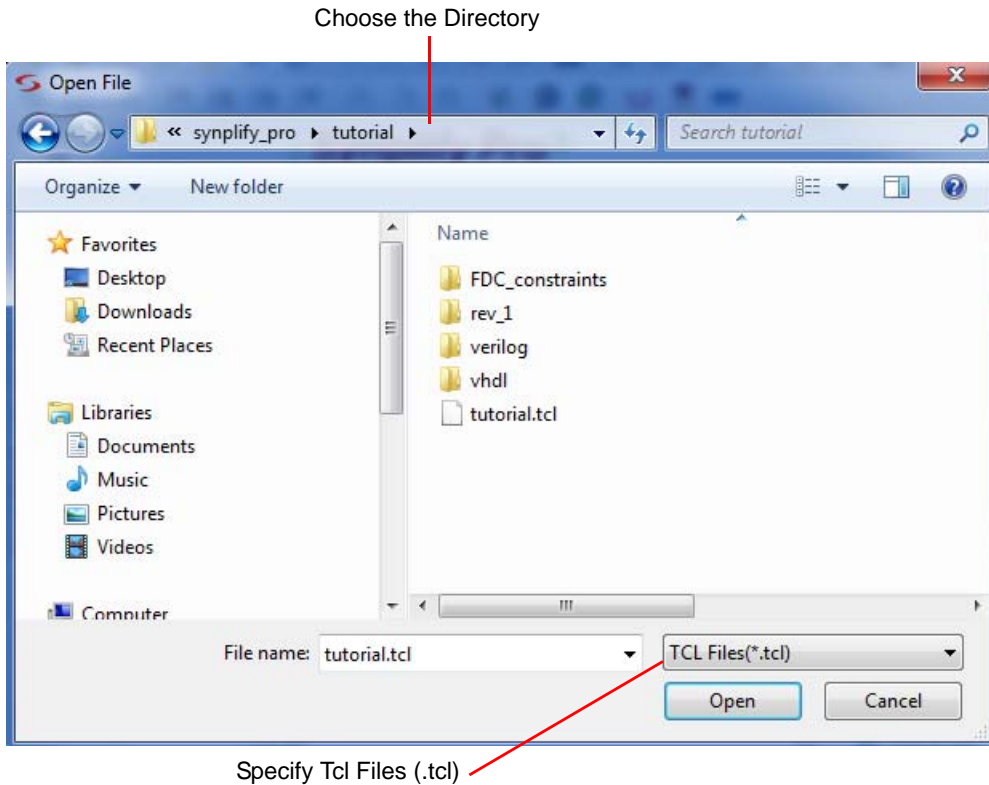
Command	Description
Launch Identify Instrumentor	Launches the Identify Instrumentor within the FPGA synthesis tools. See Working with the Identify Tools, on page 552 to launch the debugger tools.
Launch Identify Debugger	<p>Launches the Identify debugger tool. For more information, see: Working with the Identify Tools, on page 552 of the <i>User Guide</i>.</p> <p>To launch the Identify debugger in batch mode, use the <code>set_option -identify_debug_mode 1</code> Tcl command to create an Identify implementation.</p>
Launch SYNCore	<p>Opens the Synopsys FPGA IP Core Wizard. This tool helps you build IP blocks such as memory or FIFO models for your design.</p> <p>See the Launch SYNCore Command, on page 252 for details.</p>
Configure and Launch VCS Simulator	Allows you to configure and launch the VCS simulator. See Configure and Launch VCS Simulator Command, on page 281 .
Run Tcl Script	Displays the Open dialog box, where you choose a Tcl script to run. See Run Tcl Script Command, on page 247 .
Run Implementations Setup	<p>Runs all selected implementations for one project at the same time. See Run Implementations Setup Command, on page 248.</p> <p>Tcl equivalent: run -impl "implementation1 implementation2..." -parallel</p>
Job Status	During compilation, tells you the name of the current job, and gives you the runtime and directory location of your design. This option is enabled during synthesis. See Job Status Command, on page 250 . Clicking in the status area of the Project view is a shortcut for this command.

Command	Description
Next Error/Warning	Shows the next error or warning in your source code file.
Previous Error/Warning	Shows the previous error or warning in your source code file.
Log File Message Filter	Allows messages in the current session to be elevated in severity (for example, promoted to an error from a warning), lowered in severity (for example, demoting a warning to a note), or suppressed from the log file after the next run through the Log File Filter dialog box. For more information, see Log File Message Controls, on page 204 .

Run Tcl Script Command

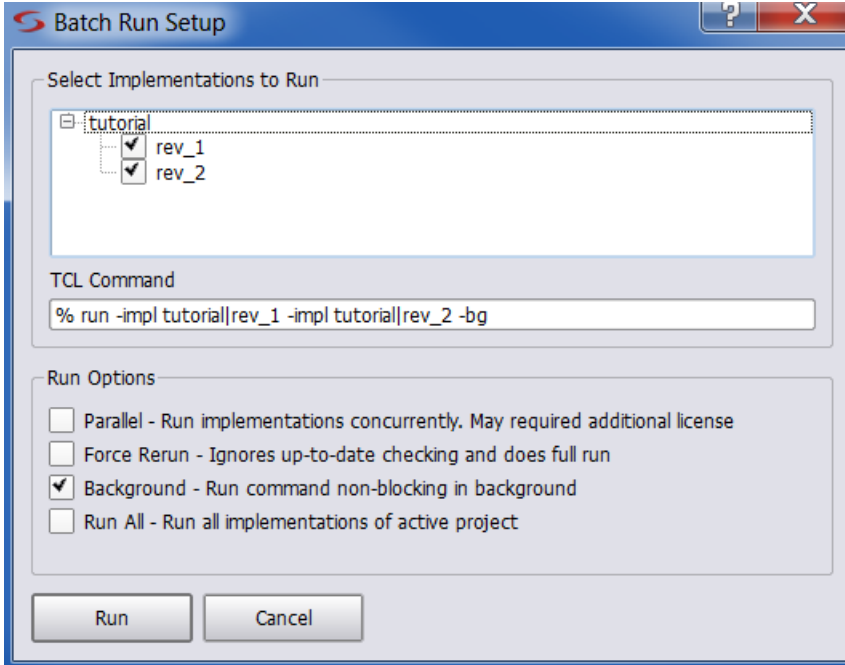
Select Run->Run Tcl Script to display the Open dialog box, where you specify the Tcl script file to execute. The File name area is filled automatically with the wildcard string “*.tcl”, corresponding to Tcl files.

This dialog box is the same as that displayed with File->Open, except that no Open as read-only check box is present. See [Open Project Command, on page 176](#), for an explanation of the features in the Open dialog box.



Run Implementations Setup Command

Select Run->Run Implementations Setup to run selected implementations of a Project file in batch mode. To use the Batch Run Setup dialog box, check one or more implementations from the list displayed and click the Run button.



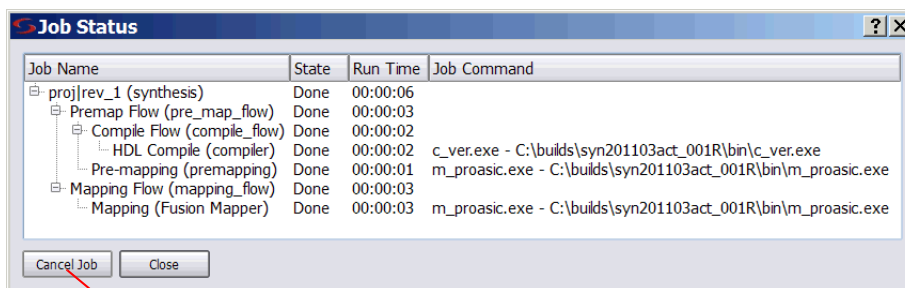
You can also choose to run the selected implementations with one or more of the following options:

Command	Description
Parallel	Runs specified implementations concurrently. This may require additional licenses. Tcl equivalent: run -impl implName -parallel
Force Rerun	Runs specified implementations, while ignoring up-to-date checking. This option clears all previous results and forces a complete rerun. Tcl equivalent: run -impl implName -clean
Background	Runs specified implementations in non-blocking background mode. Tcl equivalent: run -impl implName -bg
Run All	Runs all implementations of the active project. Tcl equivalent: run -all

Job Status Command

Select Run->Job Status to monitor the synthesis jobs that are running, their run times, and their associated commands. This information appears in the Job Status dialog box. This dialog box is also displayed when you click in the status area of the Project view (see [The Project View, on page 30](#)).

You can cancel a displayed job by selecting it in the dialog box and clicking Cancel Job.



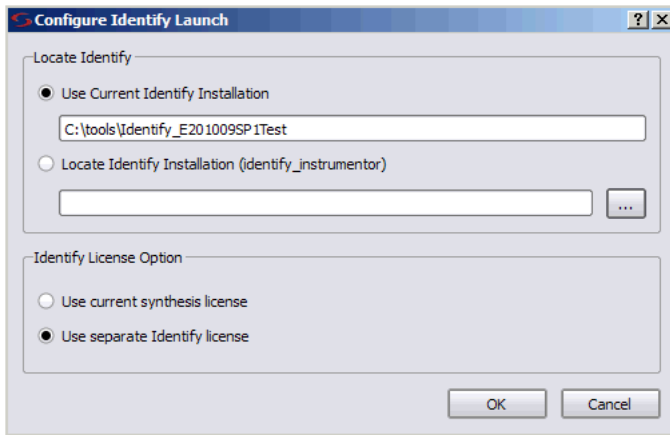
To cancel a job, select it,
then click the Cancel button

Identify Instrumentor Command

The Identify Instrumentor command lets you start the integrated or stand-alone Identify instrumentor from within the synthesis interface. Before you can use this command, you must define an Identify implementation in the project view. For a description of the work flow using the Identify instrumentor, see [Working with the Identify Tools, on page 552](#) in the *User Guide*.

Configure Identify Launch Dialog Box

Select Options->Configure Identify Launch to display this dialog box or which is automatically displayed when the location of the Identify executable has not been previously defined.



Command

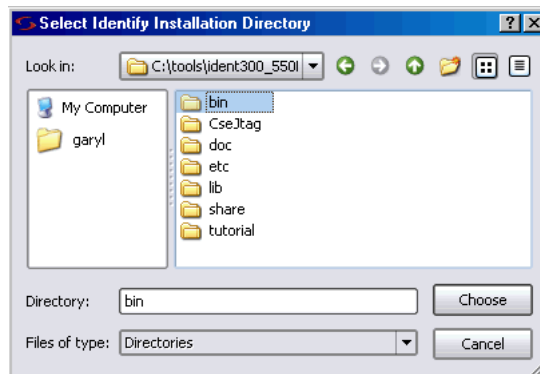
Description

Select Instrumentor

Click the radio button to select which version of the instrumentor to use. You can choose either the integrated or stand-alone Identify Instrumentor.

Locate Identify Installation (for the Identify Debugger)

A pointer to the Identify install directory. Use the (...) button to navigate to the directory location.



Identify License Option

Radio buttons to select the Identify license option. Select Use current synthesis license when only a single TSL license is available; select Use separate Identify Instrumentor license when multiple licenses are available. With a single TSL license, you are prohibited from compiling or mapping in the synthesis tool while the Identify instrumentor is open.

Launch Identify Debugger Command

The Launch Identify Debugger command launches a stand-alone version the Identify Debugger software from the synthesis interface. Before you can use this command, you must have an active Identify implementation and an instrumented design. For a description of the work flow using the Identify/Identify RTL Debugger software, see [Working with the Identify Tools, on page 552](#) in the *User Guide*.

Launch SYNCore Command

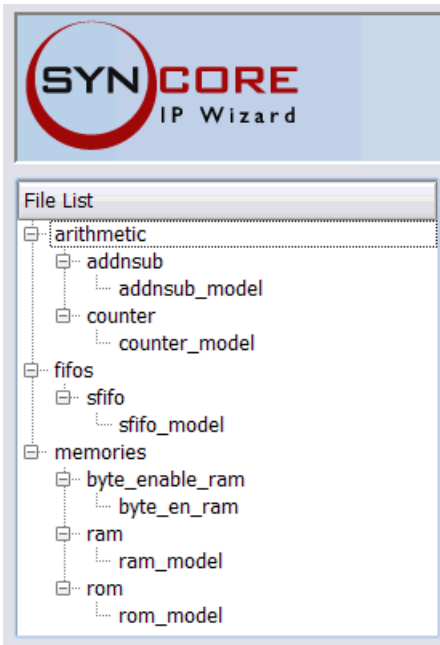
The SYNCore wizard helps you build IP cores. The wizard can compile RAM and ROM memories including a byte-enable RAM, a FIFO, an adder/subtractor, and a counter. The resulting Verilog models can be synthesized and simulated. For details about using the wizard to build these models, see the following sections in the user guide:

- [Specifying FIFOs with SYNCore, on page 450](#)
- [Specifying RAMs with SYNCore, on page 456](#)
- [Specifying Byte-Enable RAMs with SYNCore, on page 464](#)
- [Specifying ROMs with SYNCore, on page 470](#)
- [Specifying Adder/Subtractors with SYNCore, on page 475](#)
- [Specifying Counters with SYNCore, on page 482](#)

To start the SYNCore wizard, select Run->Launch SYNCore and:

- Select `sfifo_model` and click Ok to start the FIFO wizard, described in [SYNCore FIFO Wizard, on page 254](#).
- Select `ram_model` and click Ok to start the RAM wizard, described in [SYNCore RAM Wizard, on page 264](#).
- Select `byte_en_ram` and click Ok to start the byte-enable RAM wizard, described in [SYNCore Byte-Enable RAM Wizard, on page 268](#).
- Select `rom_model` and click Ok to start the ROM wizard, described in [SYNCore ROM Wizard, on page 271](#).
- Select `addnsub_model` and click Ok to start the adder/subtractor wizard, described in [SYNCore Adder/Subtractor Wizard, on page 275](#).

- Select `counter_model` and click Ok to start the counter wizard, described in [SYNCore Counter Wizard, on page 279](#).



Each SYNCore wizard has three tabs at the top, and buttons below, which are described here:

Parameters	Consists of a multiple-screen wizard that lets you set parameters for that model. See SYNCore FIFO Wizard, on page 254 , SYNCore RAM Wizard, on page 264 , SYNCore Byte-Enable RAM Wizard, on page 268 , SYNCore ROM Wizard, on page 271 , SYNCore Adder/Subtractor Wizard, on page 275 , or SYNCore Counter Wizard, on page 279 for details about the options you can set.
Core Overview	Contains basic information about the kind of model you are creating.

Generate	Generates the model with the parameters you specify in the wizard.
Sync FIFO Info, RAM Info, BYTE ENABLE RAM Info, ROM Info, ADDnSUB Info, COUNTER Info	Opens a window with technical information about the corresponding model.

SYNCore FIFO Wizard

The following describe the parameters you can set in the FIFO wizard, which opens when you select `sfifo_model`:

- [SYNCore FIFO Parameters Page 1, on page 254](#)
- [SYNCore FIFO Parameters Page 2, on page 256](#)
- [SYNCore FIFO Parameters Page 3, on page 258](#)
- [SYNCore FIFO Parameters Page 4, on page 260](#)
- [SYNCore FIFO Parameters Page 5, on page 262](#)

SYNCore FIFO Parameters Page 1

The page 1 parameters define the FIFO. Data is written/read on the rising edge of the clock.

Sync Fifo Compiler

Component Name

Directory

Filename

Sync FIFO Size

Width Valid Range 1..256

Depth Valid Range 8..16384

Parameter	Function
Component Name	Specifies a name for the FIFO. This is the name that you instantiate in your design file to create an instance of the SYNCORE FIFO in your design. Do not use spaces.
Directory	Indicates the directory where the generated files will be stored. Do not use spaces.
Filename	Specifies the name of the generated file containing the HDL description of the generated FIFO. Do not use spaces.
Width	Specifies the width of the FIFO data input and output. It must be within the valid range.
Depth	Specifies the depth of the FIFO. It must be within the valid range.

SYNCore FIFO Parameters Page 2

Sync Fifo Compiler

Sync FIFO Optional Flags

Write Control Handshaking Options

Full Flags

☒ Full Flag

☒ Active High ☐ Active Low

☐ Almost Full Flag

☒ Active High ☐ Active Low

Overflow

☐ Overflow Flag

☒ Active High ☐ Active Low

Write Acknowledge

☐ Write Acknowledge Flag

☒ Active High ☐ Active Low

The page 2 parameters let you specify optional handshaking flags for FIFO write operations. When you specify a flag, the symbol on the left reflects your choice. Data is written/read on the rising edge of the clock.

Parameter	Function
Full Flag	<p>Specifies a Full signal, which is asserted when the FIFO memory queue is full and no more writes can be performed until data is read.</p> <p>Enabling this option makes the Active High and Active Low options (FULL_FLAG_SENSE parameter) available for the signal. See Full/Almost Full Flags, on page 311 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>
Almost Full Flag	<p>Specifies an Almost_full signal, which is asserted to indicate that there is one location left and the FIFO will be full after one more write operation.</p> <p>Enabling this option makes the Active High and Active Low options available for the signal (AFULL_FLAG_SENSE parameter). See Full/Almost Full Flags, on page 311 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>
Overflow Flag	<p>Specifies an Overflow signal, which is asserted to indicate that the write operation was unsuccessful because the FIFO was full.</p> <p>Enabling this option makes the Active High and Active Low options available for the signal (OVERFLOW_FLAG_SENSE parameter). See Handshaking Flags, on page 313 f and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>
Write Acknowledge Flag	<p>Specifies a Write_ack signal, which is asserted at the completion of a successful write operation.</p> <p>Enabling this option makes the Active High and Active Low options (WACK_FLAG_SENSE parameter) available for the signal. See Handshaking Flags, on page 313 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>
Active High	Sets the specified signal to active high (1).
Active Low	Sets the specified signal to active low (0).

SYNCore FIFO Parameters Page 3

The page 3 parameters let you specify optional handshaking flags for FIFO read operations. Data is written/read on the rising edge of the clock.

Sync Fifo Compiler

Sync FIFO Optional Flags

Read Control Handshaking Options

Empty Flag

☒ Empty Flag

☒ Active High
☐ Active Low

☐ Almost Empty Flag

☒ Active High
☐ Active Low

Underflow

☐ Underflow Flag

☒ Active High
☐ Active Low

Read Acknowledge

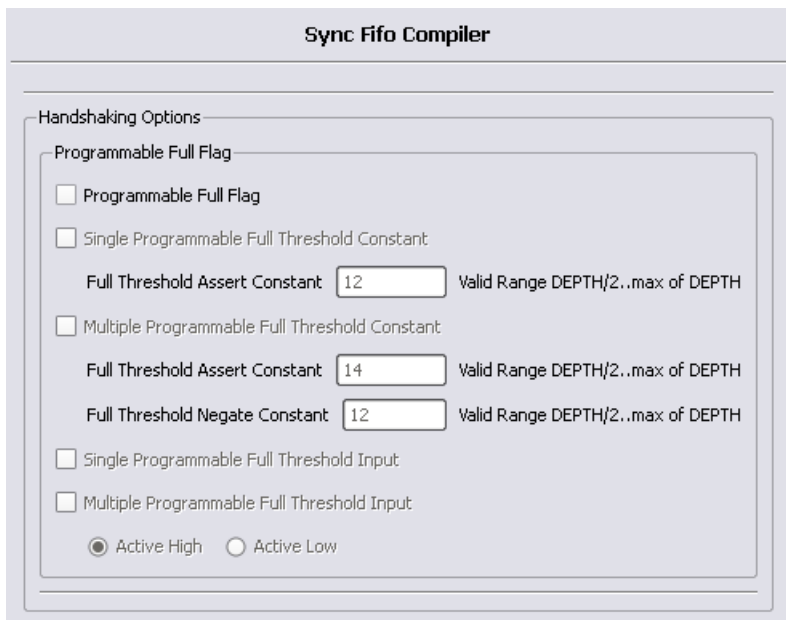
☐ Read Acknowledge Flag

☒ Active High
☐ Active Low

Parameter	Function
Empty Flag	<p>Specifies an Empty signal, which is asserted when the memory queue for the FIFO is empty and no more reads can be performed until data is written.</p> <p>Enabling this option makes the Active High and Active Low options (EMPTY_FLAG_SENSE parameter) available for the signal. See Empty/Almost Empty Flags, on page 312 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>

Parameter	Function
Almost Empty Flag	<p>Specifies an Almost_empty signal, which is asserted when there is only one location left to be read. The FIFO will be empty after one more read operation.</p> <p>Enabling this option makes the Active High and Active Low options (AEMPTY_FLAG_SENSE parameter) available for the signal. See Empty/Almost Empty Flags, on page 312 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>
Underflow Flag	<p>Specifies an Underflow signal, which is asserted to indicate that the read operation was unsuccessful because the FIFO was empty.</p> <p>Enabling this option makes the Active High and Active Low options (UNDRFLW_FLAG_SENSE parameter) available for the signal. See Handshaking Flags, on page 313 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>
Read Acknowledge Flag	<p>Specifies a Read_ack signal, which is asserted at the completion of a successful read operation.</p> <p>Enabling this option makes the Active High and Active Low options (RACK_FLAG_SENSE parameter) available for the signal. See Handshaking Flags, on page 313 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>
Active High	Sets the specified signal to active high (1).
Active Low	Sets the specified signal to active low (0).

SYNCore FIFO Parameters Page 4



Sync Fifo Compiler

Handshaking Options

Programmable Full Flag

☐ Programmable Full Flag

☐ Single Programmable Full Threshold Constant

Full Threshold Assert Constant: Valid Range DEPTH/2..max of DEPTH

☐ Multiple Programmable Full Threshold Constant

Full Threshold Assert Constant: Valid Range DEPTH/2..max of DEPTH

Full Threshold Negate Constant: Valid Range DEPTH/2..max of DEPTH

☐ Single Programmable Full Threshold Input

☐ Multiple Programmable Full Threshold Input

☒ Active High ☐ Active Low

The page 4 parameters let you specify optional handshaking flags for FIFO programmable full operations. To use these options, you must have a Full signal specified. See [FIFO Programmable Flags, on page 314](#) for details and [FIFO Parameters, on page 309](#) for a list of the FIFO parameters. Data is written/read on the rising edge of the clock.

Parameter	Function
Programmable Full Flag	<p>Specifies a Prog_full signal, which indicates that the FIFO has reached a user-defined full threshold.</p> <p>You can only enable this option if you set Full Flag on page 2. When it is enabled, you can specify other options for the Prog_Full signal (PFULL_FLAG_SENSE parameter). See Programmable Full, on page 315 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p>

Parameter	Function
Single Programmable Full Threshold Constant	<p>Specifies a Prog_full signal with a single constant defining the assertion threshold (PGM_FULL_TYPE=1 parameter). See Programmable Full with Single Threshold Constant, on page 315 for details.</p> <p>Enabling this option makes Full Threshold Assert Constant available.</p>
Multiple Programmable Full Threshold Constant	<p>Specifies a Prog_full signal (PGM_FULL_TYPE=2 parameter), with multiple constants defining the assertion and de-assertion thresholds. See Programmable Full with Multiple Threshold Constants, on page 316 for details.</p> <p>Enabling this option makes Full Threshold Assert Constant and Full Threshold Negate Constant available.</p>
Full Threshold Assert Constant	<p>Specifies a constant that is used as a threshold value for asserting the Prog_full signal. It sets the PGM_FULL_THRESH parameter for PGM_FULL_TYPE=1 and the PGM_FULL_ATHRESH parameter for PGM_FULL_TYPE=2.</p>
Full Threshold Negate Constant	<p>Specifies a constant that is used as a threshold value for de-asserting the Prog_full signal (PGM_FULL_NTHRESH parameter).</p>
Single Programmable Full Threshold Input	<p>Specifies a Prog_full signal (PGM_FULL_TYPE=3 parameter), with a threshold value specified dynamically through a Prog_full_thresh input port during the reset state. See Programmable Full with Single Threshold Input, on page 316 for details.</p> <p>Enabling this option adds the Prog_full_thresh input port to the FIFO.</p>
Multiple Programmable Full Threshold Input	<p>Specifies a Prog_full signal (PGM_FULL_TYPE=4 parameter), with threshold assertion and deassertion values specified dynamically through input ports during the reset state. See Programmable Full with Multiple Threshold Inputs, on page 317 for details.</p> <p>Enabling this option adds the Prog_full_thresh_assert and Prog_full_thresh_negate input ports to the FIFO.</p>
Active High	Sets the specified signal to active high (1).
Active Low	Sets the specified signal to active low (0).

SYNCore FIFO Parameters Page 5

These options specify optional handshaking flags for FIFO programmable empty operations. To use these options, you first specify an Empty signal on page 3. See [FIFO Programmable Flags, on page 314](#) for details and [FIFO Parameters, on page 309](#) for a list of the FIFO parameters. Data is written/read on the rising edge of the clock.

The screenshot shows the 'Sync Fifo Compiler' window. It has two main sections: 'Handshaking Options' and 'Number of Words in FIFO'. The 'Handshaking Options' section contains a 'Programmable Empty Flag' group with several options: 'Programmable Empty Flag' (unchecked), 'Single Programmable Empty Threshold Constant' (unchecked), 'Empty Threshold Assert Constant' (set to 4, with a valid range of 1..max of DEPTH/2), 'Multiple Programmable Empty Threshold Constant' (unchecked), 'Empty Threshold Assert Constant' (set to 2, with a valid range of 1..max of DEPTH/2), 'Empty Threshold Negate Constant' (set to 4, with a valid range of 1..max of DEPTH/2), 'Single Programmable Empty Threshold Input' (unchecked), and 'Multiple Programmable Empty Threshold Input' (unchecked). Below these are radio buttons for 'Active High' (selected) and 'Active Low'. The 'Number of Words in FIFO' section contains a checkbox for 'Number of valid Data in Fifo' which is unchecked.

Parameter	Function
Programmable Empty Flag	<p>Specifies a Prog_empty signal (PEMPTY_FLAG_SENSE parameter), which indicates that the FIFO has reached a user-defined empty threshold. See Programmable Empty, on page 318 and FIFO Parameters, on page 309 for descriptions of the flag and parameter.</p> <p>Enabling this option makes the other options available to specify the threshold value, either as a constant or through input ports. You can also specify single or multiple thresholds for each of these options.</p>

Parameter	Function
Single Programmable Empty Threshold Constant	<p>Specifies a Prog_empty signal (PGM_EMPTY_TYPE=1 parameter), with a single constant defining the assertion threshold. See Programmable Empty with Single Threshold Input, on page 319 for details.</p> <p>Enabling this option makes Empty Threshold Assert Constant available.</p>
Multiple Programmable Empty Threshold Constant	<p>Specifies a Prog_empty signal (PGM_EMPTY_TYPE=2 parameter), with multiple constants defining the assertion and de-assertion thresholds. See Programmable Empty with Multiple Threshold Constants, on page 319 for details.</p> <p>Enabling this option makes Empty Threshold Assert Constant and Empty Threshold Negate Constant available.</p>
Empty Threshold Assert Constant	<p>Specifies a constant that is used as a threshold value for asserting the Prog_empty signal. It sets the PGM_EMPTY_THRESH parameter for PGM_EMPTY_TYPE=1 and the PGM_EMPTY_ATHRESH parameter for PGM_EMPTY_TYPE=2.</p>
Empty Threshold Negate Constant	<p>Specifies a constant that is used as a threshold value for de-asserting the Prog_empty signal (PGM_EMPTY_NTHRESH parameter).</p>
Single Programmable Empty Threshold Input	<p>Specifies a Prog_empty signal (PGM_EMPTY_TYPE=3 parameter), with a threshold value specified dynamically through a Prog_empty_thresh input port during the reset state. See Programmable Empty with Single Threshold Input, on page 319 for details.</p> <p>Enabling this option adds the Prog_full_thresh input port to the FIFO.</p>
Multiple Programmable Empty Threshold Input	<p>Specifies a Prog_empty signal (PGM_EMPTY_TYPE=4 parameter), with threshold assertion and deassertion values specified dynamically through Prog_empty_thresh_assert and Prog_empty_thresh_negate input ports during the reset state. See Programmable Empty with Multiple Threshold Inputs, on page 320 for details.</p> <p>Enabling this option adds the input ports to the FIFO.</p>
Active High	Sets the specified signal to active high (1).
Active Low	Sets the specified signal to active low (0).

Parameter	Function
Number of Valid Data in FIFO	Specifies the Data_cnt signal for the FIFO output. This signal contains the number of words in the FIFO in the read domain.

SYNCore RAM Wizard

The following describe the parameters you can set in the RAM wizard, which opens when you select ram_model:

- [SYNCore RAM Parameters Page 1, on page 264](#)
- [SYNCore RAM Parameters Pages 2 and 3, on page 266](#)

SYNCore RAM Parameters Page 1

The screenshot displays the 'Memory Compiler' wizard interface. It includes the following fields and options:

- Component Name:** A text input field.
- Directory:** A text input field with a 'Browse...' button.
- Filename:** A text input field with a 'Browse...' button.
- Memory Size:** A section containing:
 - Data Width:** A text input field with the value '16' and a 'Valid Range 1..256' label.
 - Address Width:** A text input field with the value '8' and a 'Valid Range 2..256' label.
- How will you be using the RAM?:** A section with two radio buttons: 'Single Port' (selected) and 'Dual Port'.
- Which clocking method do you want to use?:** A section with two radio buttons: 'Single Clock' (selected) and 'Separate Clocks For Each Port'.

Component Name	<p>Specifies the name of the component. This is the name that you instantiate in your design file to create an instance of the SYNCORE RAM in your design. Do not use spaces. For example:</p> <pre> ram101 <ComponentName> (.PortAClk (PortAClk) , .PortAAddr (PortAAddr) , .PortADataIn (PortADataIn) , .PortAWriteEnable (PortAWriteEnable) , .PortBDataIn (PortBDataIn) , .PortBAddr (PortBAddr) , .PortBWriteEnable (PortBWriteEnable) , .PortADataOut (PortADataOut) , .PortBDataOut (PortBDataOut)); </pre>
Directory	<p>Specifies the directory where the generated files are stored. Do not use spaces. The following files are created:</p> <ul style="list-style-type: none"> • <code>filelist.txt</code> – lists files written out by SYNCORE • <code>options.txt</code> – lists the options selected in SYNCORE • <code>readme.txt</code> – contains a brief description and known issues • <code>syncore_ram.v</code> – Verilog library file required to generate RAM model • <code>testbench.v</code> – Verilog testbench file for testing the RAM model • <code>instantiation_file.vin</code> – describes how to instantiate the wrapper file • <code>component.v</code> – RAM model wrapper file generated by SYNCORE <p>Note that running the Memory Compiler wizard in the same directory overwrites the existing files.</p>
Filename	Specifies the name of the generated file containing the HDL description of the compiled RAM. Do not use spaces.
Data Width	Is the width of the data you need for the memory. The unit used is the number of bits.
Address Width	Is the address depth you need for the memory. The unit used is the number of bits.
Single Port	When enabled, generates a single-port RAM.

Dual Port	When enabled, generates a dual-port RAM.
Single Clock	When enabled, generates a RAM with a single clock for dual-port configurations.
Separate Clocks for Each Port	When enabled, generates separate clocks for each port in dual-port RAM configurations.

SYNCore RAM Parameters Pages 2 and 3

The port implementation parameters on pages 2 and 3 are identical, but page 2 applies to Port A (single- and dual-port configurations), and page 3 applies to Port B (dual-port configurations only). The following figure shows the parameters on page 2 for Port A.

Memory Compiler

Configuring Port A

How do you want to configure Port A

☒ Read And Write Access ☐ Read Only Access ☐ Write Only Access

Design Options for Port A

☒ Use Write Enable

☒ Register Read Address

☒ Register Outputs

☐ Synchronous Reset

Read Access Options for Port A

☒ Read before Write ☐ Read after Write ☐ No Read on Write

Read and Write Access	Specifies that the port can be accessed by both read and write operations.
Read Only Access	Specifies that the port can only be accessed by read operations.
Write Only Access	Specifies that the port can only be accessed by write operations.
Use Write Enable	Includes write-enable control. The RAM symbol on the left reflects the selections you make.

Register Read Address	Adds registers to the read address lines. The RAM symbol on the left reflects the selections you make.
Register Outputs	Adds registers to the write address lines when you specify separate read/write addressing. The register outputs are always enabled. The RAM symbol on the left reflects the selections you make.
Synchronous Reset	Individually synchronizes the reset signal with the clock when you enable Register Outputs. The RAM symbol on the left reflects the selections you make.
Read before Write	Specifies that the read operation takes place before the write operation for port configurations with both read and write access (Read And Write Access is enabled). For a timing diagram, see Read Before Write, on page 328 .
Read after Write	Specifies that the read operation takes place after the write operation for port configurations with both read and write access (Read And Write Access is enabled). For a timing diagram, see Write Before Read, on page 329 .
No Read on Write	Specifies that no read operation takes place when there is a write operation for port configurations with both read and write access (Read And Write Access is enabled). For a timing diagram, see No Read on Write, on page 330 .

SYNCore Byte-Enable RAM Wizard

The following describes the parameters you can set in the byte-enable RAM wizard, which opens when you select `byte_en_ram`.

- [SYNCore Byte-Enable RAM Parameters Page 1, on page 268](#)
- [SYNCore Byte-Enable RAM Parameters Pages 2 and 3, on page 269](#)

SYNCore Byte-Enable RAM Parameters Page 1

The screenshot shows the 'Byte Enable Ram Compiler' wizard. It contains several input fields and a section for selecting the RAM configuration.

Byte Enable Ram Compiler

Component Name:

Directory:

File Name:

Memory Size

Address Width: Valid Range 1...256

Data Width: valid range 1..256

Write Enable Width: Valid Range 1...256

How will you be using the RAM?

☒ Single Port ☐ Dual Port

Component Name	Specifies the name of the component. This is the name that you instantiate in your design file to create an instance of the SYNCORE byte-enable RAM in your design. Do not use spaces.
Directory	<p>Specifies the directory where the generated files are stored. Do not use spaces. The following files are created:</p> <ul style="list-style-type: none"> • filelist.txt – lists files written out by SYNCORE • options.txt – lists the options selected in SYNCORE • readme.txt – contains a brief description and known issues • syncore_be_ram_sdp.v – SystemVerilog library file required to generate single or simple dual-port, byte-enable RAM model • syncore_be_ram_tdp.v – SystemVerilog library file required to generate true dual-port byte-enable RAM model • testbench.v – Verilog testbench file for testing the byte-enable RAM model • instantiation_file.vin – describes how to instantiate the wrapper file • <i>component.v</i> – Byte-enable RAM model wrapper file generated by SYNCORE <p>Note that running the byte-enable RAM wizard in the same directory overwrites the existing files.</p>
Filename	Specifies the name of the generated file containing the HDL description of the compiled byte-enable RAM. Do not use spaces.
Address Width	Specifies the address depth for Ports A and B. The unit used is the number of bits; the default is 2.
Data Width	Specifies the width of the data for Ports A and B. The unit used is the number of bits; the default is 2.
Write Enable Width	Specifies the write enable width for Ports A and B. The unit used is the number of byte enables; the default is 2, the maximum is 4.
Single Port	When enabled, generates a single-port, byte-enable RAM (automatically enables single clock).
Dual Port	When enabled, generates a dual-port, byte-enable RAM (automatically enables separate clocks for each port).

SYNCore Byte-Enable RAM Parameters Pages 2 and 3

The port implementation parameters on pages 2 and 3 are identical, but page 2 applies to Port A (single- and dual-port configurations), and page 3 applies to Port B (dual-port configurations only). The following figure shows the parameters on page 2 for Port A.

Byte Enable Ram Compiler

Configuring Port A

How do you want to configure Port A

☒ Read And Write Access
 ☐ Read Only Access
 ☐ Write Only Access

Pipelining Address Bus and Output Data

☒ Register address bus AddrA
☒ Register output data bus RdDataA

Configure Reset Options

☒ Reset for RdDataA

Specify output data on reset

☒ Default value of '1' for all bits
☐ Specify Reset value for RdDataA Valid Range 0...2^{DATA_WIDTH}

Configure Write Enable Options

☒ Write Enable for PORTA
☒ Active High
 ☐ Active Low

Read and Write Access	Specifies that the port can be accessed by both read and write operations (only mode allowed for single-port configurations).
Read Only Access	Specifies that the port can only be accessed by read operations (dual-port mode only).
Write Only Access	Specifies that the port can only be accessed by write operations (dual-port mode only).
Register address bus AddrA/B	Adds registers to the read address lines.
Register output data bus RdDataA/B	Adds registers to the read data lines. By default, the read data register is enabled.

Reset for RdDataA/B	<p>Specifies the reset type for registered read data:</p> <ul style="list-style-type: none"> Reset type is synchronous when Reset for RdDataA/B is enabled Reset type is no reset when Reset for RdDataA/B is disabled
Specify output data on reset	<p>Specifies reset value for registered read data (applies only when RdDataA/B is enabled):</p> <ul style="list-style-type: none"> Default value of '1' for all bits – sets read data to all 1's on reset Specify Reset value for RdDataA/B – specifies reset value for read data; when enabled, value is entered in adjacent field
Write Enable for Port A/B	<p>Specifies the write enable level for Port A/B. Default is Active High.</p>

SYNCore ROM Wizard

The following describe the parameters you can set in the ROM wizard, which opens when you select rom_model:

- [SYNCore ROM Parameters Page 1, on page 272](#)
- [SYNCore ROM Parameters Pages 2 and 3, on page 273](#)
- [SYNCore ROM Parameters Page 4, on page 275](#)

SYNCore ROM Parameters Page 1

Component Name:

Directory:

File Name:

ROM Size

Read Data width: Valid Range 1..256

ROM address width: Valid Range 2..256

Configuring the ROM

☒ Single Port Rom ☐ Dual Port Rom

Component Name	Specifies the name of the component. This is the name that you instantiate in your design file to create an instance of the SYNCore ROM in your design. Do not use spaces.
Directory	<p>Specifies the directory where the generated files are stored. Do not use spaces. The following files are created:</p> <p>filelist.txt – lists files written out by SYNCore</p> <p>options.txt – lists the options selected in SYNCore</p> <p>readme.txt – contains a brief description and known issues</p> <p>syncore_rom.v – Verilog library file required to generate ROM model</p> <p>testbench.v – Verilog testbench file for testing the ROM model</p> <p>instantiation_file.vin – describes how to instantiate the wrapper file</p> <p><i>component.v</i> – ROM model wrapper file generated by SYNCore</p> <p>Note that running the ROM wizard in the same directory overwrites the existing files.</p>
File Name	Specifies the name of the generated file containing the HDL description of the compiled ROM. Do not use spaces.

Read Data Width	Specifies the read data width of the ROM. The unit used is the number of bits and ranges from 2 to 256. Default value is 8. The read data width is common to both Port A and Port B. The corresponding file parameter is DATA_WIDTH=n.
ROM address width	Specifies the address depth for the memory. The unit used is the number of bits. Default value is 10. The corresponding file parameter is ADD_WIDTH=n.
Single Port Rom	When enabled, generates a single-port ROM. The corresponding file parameter is CONFIG_PORT="single".
Dual Port Rom	When enabled, generates a dual-port ROM. The corresponding file parameter is CONFIG_PORT="dual".

SYNCore ROM Parameters Pages 2 and 3

The port implementation parameters on pages 2 and 3 are the same; page 2 applies to Port A (single- and dual-port configurations), and page 3 applies to Port B (dual-port configurations only).

Configuring Port A

Pipelining Address Bus and Output Data

☐ Register address bus AddrA
 ☒ Register output data bus DataA

Configure Reset Options

☒ Reset for PORTA

☒ Asynchronous Reset
 ☐ Synchronous Reset

Configure Enable

☒ Enable for PORTA

☒ Active High Enable
 ☐ Active Low Enable

Specify output data on reset

☒ Default value of '1' for all bits
 ☐ Specify reset value for DataA Valid Range 0...2^{DATA_WIDTH}

Register address bus AddrA	Used with synchronous ROM configurations to register the read address. When checked, also allows chip enable to be configured.
Register output data bus DataA	Used with synchronous ROM configurations to register the data outputs. When checked, also allows chip enable to be configured.
Asynchronous Reset	Sets the type of reset to asynchronous (Configure Reset Options must be checked). Configuring reset also allows the output data pattern on reset to be defined. The corresponding file parameter is RST_TYPE_A=1/RST_TYPE_B=1.
Synchronous Reset	Sets the type of reset to synchronous (Configure Reset Options must be checked). Configuring reset also allows the output data pattern on reset to be defined. The corresponding file parameter is RST_TYPE_A=0/RST_TYPE_B=0.
Active High Enable	Sets the level of the chip enable to high for synchronous ROM configurations. The corresponding file parameter is EN_SENSE_A=1/EN_SENSE_B=1.
Active Low Enable	Sets the level of the chip enable to low for synchronous ROM configurations. The corresponding file parameter is EN_SENSE_A=0/EN_SENSE_B=0.
Default value of '1' for all bits	Specifies an output data pattern of all 1's on reset. The corresponding file parameter is RST_DATA_A={n{1'b1} }/RST_DATA_B={n{1'b1} }.
Specify reset value for DataA/DataB	Specifies a user-defined output data pattern on reset. The pattern is defined in the adjacent field. The corresponding file parameter is RST_TYPE_A=pattern/RST_TYPE_B=pattern.

SYNCore ROM Parameters Page 4

Initialization of ROM

Select the type of the Initial Values

☒ Binary ☐ Hexadecimal

Initialization File

Binary	Specifies binary-formatted initialization file.
Hexadecimal	Specifies hexadecimal-formatted initial file.
Initialization File	Specifies path and filename of initialization file. The corresponding file parameter is INIT_FILE="filename".

SYNCore Adder/Subtractor Wizard

The following describe the parameters you can set in the adder/subtractor wizard, which opens when you select addnsub_model:

- [SYNCore Adder/Subtractor Parameters Page 1, on page 276](#)
- [SYNCore Adder/Subtractor Parameters Page 2, on page 277](#)

SYNCore Adder/Subtractor Parameters Page 1

Component Name:

Directory:

File Name:

Configure the Mode of Operation

☐ Adder

☐ Subtractor

☒ Adder/Subtractor

Component Name	Specifies a name for the adder/subtractor. This is the name that you instantiate in your design file to create an instance of the SYNCore adder/subtractor in your design. Do not use spaces.
Directory	<p>Indicates the directory where the generated files will be stored. Do not use spaces. The following files are created:</p> <ul style="list-style-type: none"> • filelist.txt – lists files written out by SYNCore • options.txt – lists the options selected in SYNCore • readme.txt – contains a brief description and known issues • syncore_ADDnSUB.v – Verilog library file required to generate adder/subtractor model • testbench.v – Verilog testbench file for testing the adder/subtractor model • instantiation_file.vin – describes how to instantiate the wrapper file • <i>component.v</i> – adder/subtractor model wrapper file generated by SYNCore <p>Note that running the wizard in the same directory overwrites any existing files.</p>
Filename	Specifies the name of the generated file containing the HDL description of the generated adder/subtractor. Do not use spaces.

Adder	When enabled, generates an adder (the corresponding file parameter is ADD_N_SUB ="ADD").
Subtractor	When enabled, generates a subtractor (the corresponding file parameter is ADD_N_SUB ="SUB").
Adder/Subtractor	When enabled, generates a dynamic adder/subtractor (the corresponding file parameter is ADD_N_SUB ="DYNAMIC").

SYNCore Adder/Subtractor Parameters Page 2

Input and Output Ports Configurations

Configure Port A

Port A Width

☒ Register Input A

☒ Clock Enable for Register A ☒ Reset for Register A

Configure Port B

☐ Constant Value Input ☒ Enable Port B

Constant Value/Port B Width

☒ Register Input B

☒ Clock Enable for Register B ☒ Reset for Register B

Configure Output Port

Output port Width

☒ Register output PortOut

☒ Clock Enable for Register PortOut ☒ Reset for Register PortOut

Configure Reset type for all Reset Signals

☐ Synchronous Reset ☒ Asynchronous Reset

Port A Width	Specifies the width of port A (the corresponding file parameter is PORT_A_WIDTH=n).
Register Input A	Used with synchronous adder/subtractor configurations to register port A. When checked, also allows clock enable and reset to be configured (the corresponding file parameter is PORTA_PIPELINE_STAGE='0' or '1').
Clock Enable for Register A	Specifies the enable for port A register.
Reset for Register A	Specifies the reset for port A register.
Constant Value Input	Specifies port B as a constant input when checked and allows you to enter a constant value in the Constant Value/Port B Width field (the corresponding file parameter is CONSTANT_PORT ='0').
Enable Port B	Specifies port B as an input when checked and allows you to enter a port B width in the Constant Value/Port B Width field (the corresponding file parameter is CONSTANT_PORT ='1').
Constant Value/Port B Width	Specifies either a constant value or port B width depending on Constant Value Input and Enable Port B selection (the corresponding file parameters are CONSTANT_VALUE= n or PORT_B_WIDTH=n).
Register Input B	Used with synchronous adder/subtractor configurations to register port B. When checked, also allows clock enable and reset to be configured (the corresponding file parameter is PORTB_PIPELINE_STAGE='0' or '1').
Clock Enable for Register B	Specifies the enable for the port B register.
Reset for Register B	Specifies the reset for the port B register.
Output port Width	Specifies the width of the output port (the corresponding file parameter is PORT_OUT_WIDTH=n).
Register output PortOut	Used with synchronous adder/subtractor configurations to register the output port. When checked, also allows clock enable and reset to be configured (the corresponding file parameter is PORTOUT_PIPELINE_STAGE='0' or '1').
Clock Enable for Register PortOut	Specifies the enable for the output port register.

Reset for Register PortOut	Specifies the reset for the output port register.
Synchronous Reset	Sets the type of reset to synchronous (the corresponding file parameter is RESET_TYPE='0').
Asynchronous Reset	Sets the type of reset to asynchronous (the corresponding file parameter is RESET_TYPE='1').

SYNCore Counter Wizard

The following describe the parameters you can set in the ROM wizard, which opens when you select counter_model:

- [SYNCore Counter Parameters Page 1, on page 279](#)
- [SYNCore Counter Parameters Page 2, on page 281](#)

SYNCore Counter Parameters Page 1

Component Name

Directory

File Name

Configure the Counter Parameters

Width of Counter

Counter Step Value

Configure the Mode of Counter

☐ Up Counter

☐ Down Counter

☒ UpDown Counter

Component Name	Specifies a name for the counter. This is the name that you instantiate in your design file to create an instance of the SYNCore counter in your design. Do not use spaces.
Directory	<p>Indicates the directory where the generated files will be stored. Do not use spaces. The following files are created:</p> <ul style="list-style-type: none"> • <code>filelist.txt</code> – lists files written out by SYNCore • <code>options.txt</code> – lists the options selected in SYNCore • <code>readme.txt</code> – contains a brief description and known issues • <code>syncore_counter.v</code> – Verilog library file required to generate counter model • <code>testbench.v</code> – Verilog testbench file for testing the counter model • <code>instantiation_file.vin</code> – describes how to instantiate the wrapper file • <code>component.v</code> – counter model wrapper file generated by SYNCore <p>Note that running the wizard in the same directory overwrites any existing files.</p>
Filename	Specifies the name of the generated file containing the HDL description of the generated counter. Do not use spaces.
Width of Counter	Determines the counter width (the corresponding file parameter is <code>COUNT_WIDTH=n</code>).
Counter Step Value	Determines the counter step value (the corresponding file parameter is <code>STEP=n</code>).
Up Counter	Specifies an up counter (the default) configuration (the corresponding file parameter is <code>MODE=Up</code>).
Down Counter	Specifies an down counter configuration (the corresponding file parameter is <code>MODE=Down</code>).
UpDown Counter	Specifies a dynamic up/down counter configuration (the corresponding file parameter is <code>MODE=Dynamic</code>).

SYNCore Counter Parameters Page 2

Additional Configurations

Configure Load options

☒ Enable Load option

Configure Load Value

☐ Load Constant Value

Load Value for constant load option

☒ Use the port PortLoadValue to load Value

Configure Reset type

☒ Synchronous Reset ☐ Asynchronous Reset

Enable Load option	Enables the load options
Load Constant Value	Load the constant value specified in the Load Value for constant load option field; (the corresponding file parameter is LOAD=1).
Load Value for constant load option	The constant value to be loaded.
Use the port PortLoadValue to load Value	Loads variable value from PortLoadValue (the corresponding file parameter is LOAD=2).
Synchronous Reset	Specifies a synchronous (the default) reset input (the corresponding file parameter is MODE=0).
Asynchronous Reset	Specifies an asynchronous reset input (the corresponding file parameter is MODE=1).

Configure and Launch VCS Simulator Command

The Configure and Launch VCS Simulator command enables you to launch VCS simulation from within the Synopsys FPGA synthesis tool. Additionally, configuration information, such as libraries and options can be specified on the Run VCS Simulator dialog box before running VCS simulation. You can launch this simulation tool from the synthesis tools on Linux platforms only.

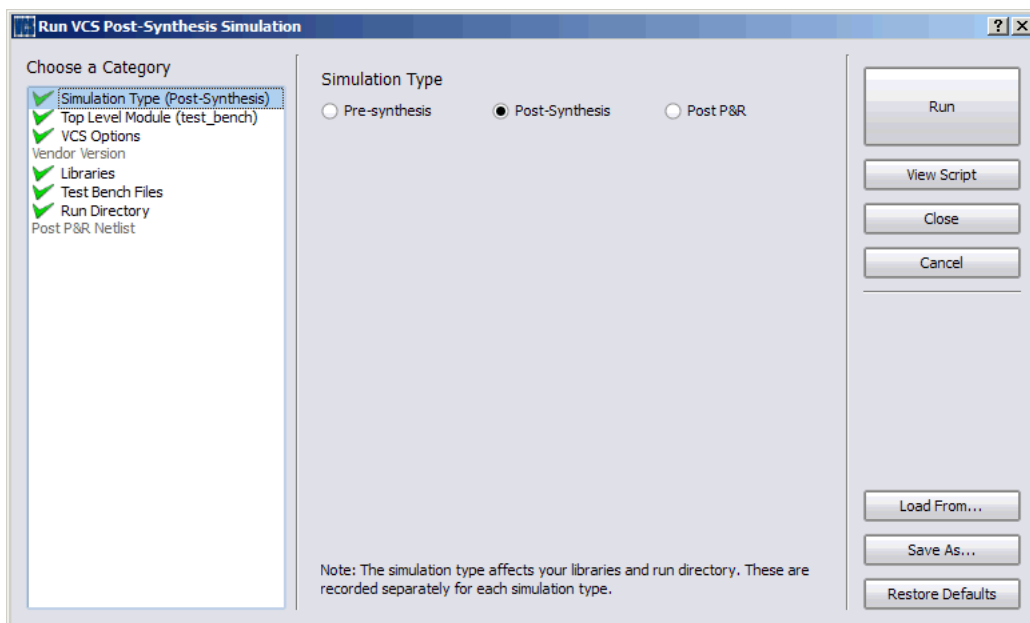
For a step-by-step procedure on setting up and launching this tool, see [Simulating with the VCS Tool, on page 560](#) in the *User Guide*.

The Run VCS *SimulationType* Simulation dialog box contains unique pages for specific tasks, such as specifying simulation type, VCS options, and libraries or test bench files. From this dialog box:

- Choose a category, which simplifies the data input for each task.
- A task marked with (✓) means that data has automatically been filled in; however, an (✗) requires that data must be filled in.
- You are prompted to save, after canceling changes made in the dialog box.

Simulation Type

The following dialog box displays the Simulation Type task.



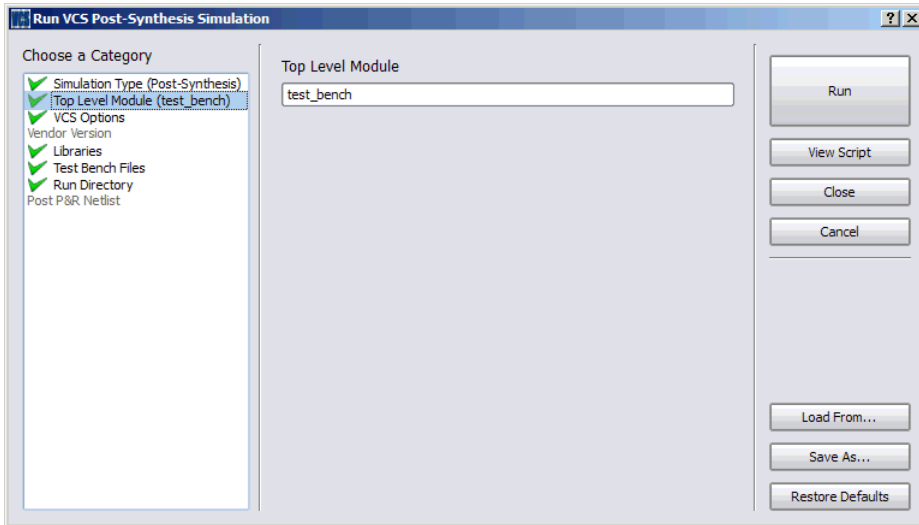
The Run VCS Simulator dialog box contains the following options:

Command	Description
Choose a Category Simulation Type	<p>Select Simulation Type and choose the type of simulation to run:</p> <ul style="list-style-type: none"> • Pre-synthesis – RTL simulation • Post-synthesis – Post-synthesis netlist simulation • Post-P&R – Post-P&R netlist simulation <p>See Simulation Type, on page 282 to view the dialog box.</p>
Choose a Category Top Level Module	<p>Select Top Level Module and specify the top-level VCS module or modules for simulation. You can use any combination of the semi-colon (;), comma (,), or a space to separate multiple top-level modules.</p> <p>See Top Level Module, on page 285 to view the dialog box.</p>
Choose a Category VCS Options	<p>Select VCS Options and specify options for each VCS step:</p> <ul style="list-style-type: none"> • Verilog compiler – VLOGAN command options for compiling and analyzing Verilog, such as the -q option. • VHDL compiler – VHDLAN options for compiling and analyzing VHDL. • Elaboration – VCS command options. The default setting is -debug_all. • Simulation – SIMV command options. The default setting is -gui. <p>The default settings use the FPGA version of VCS and open the VCS GUI for the debugger (DBE) and the waveform viewer.</p> <p>See VCS Options, on page 285 to view the dialog box.</p>
Choose a Category Vendor Version	<p>See Vendor Version, on page 286 to view the dialog box.</p>
Choose a Category Libraries	<p>Select Libraries and specify library files typically used for Post-synthesis or Post-P&R simulation. These library files are automatically populated in the display window. You can choose to:</p> <ul style="list-style-type: none"> • Add a library • Edit the selected library • Remove the selected library <p>See Libraries, on page 286 and Changing Library and Test Bench Files, on page 288 for more information.</p>

Command	Description
Choose a Category Test Bench Files	<p>Select Test Bench Files and specify the test bench files typically used for Post-synthesis or Post-P&R simulation. These test bench files are automatically populated in the display window. You can choose to:</p> <ul style="list-style-type: none"> • Add a test bench file • Edit the selected test bench file • Remove the selected test bench file <p>See Test Bench Files, on page 287 and Changing Library and Test Bench Files, on page 288 for more information.</p>
Choose a Category Run Directory	<p>Select Run Directory and specify the results directory to run the VCS simulation.</p> <p>See Run Directory, on page 287 to view the dialog box.</p>
Choose a Category Post P&R Netlist	<p>Select Post P&R Netlist and specify the post place-and-route netlist to run the VCS simulation.</p> <p>See Post P&R Netlist, on page 288 to view the dialog box.</p>
Run	Runs VCS simulation.
View Script	View the script file with the specified VCS commands and options before generating it. For an example, see VCS Script File, on page 290 .
Load From	Use this option to load an existing VCS script.
Save As	Generates the VCS script. The tool generates the XML script in the directory specified.
Restore Defaults	Restores all the default VCS settings.

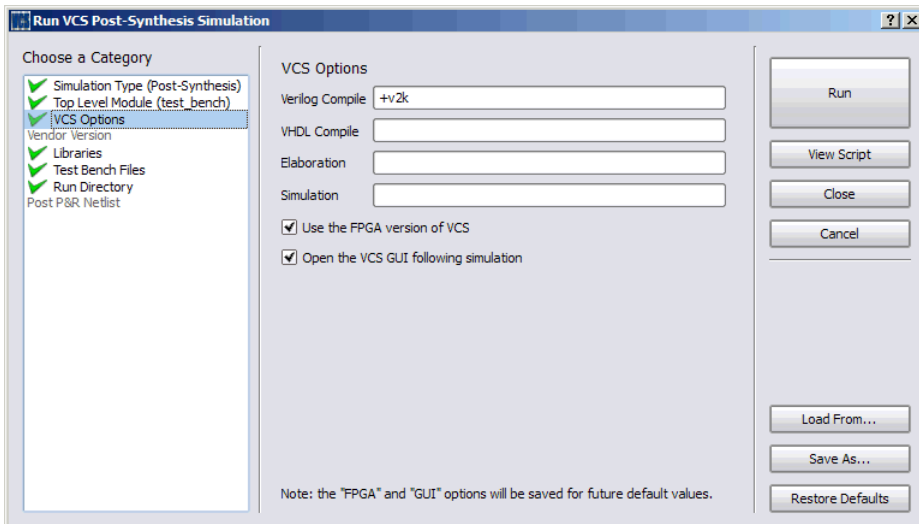
Top Level Module

The following dialog box displays the Top Level Module task.



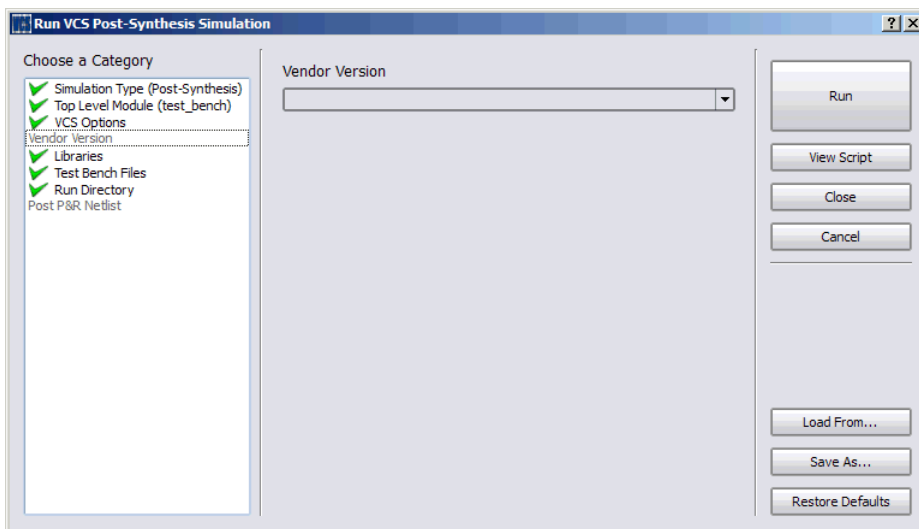
VCS Options

The following dialog box displays the VCS Options task.



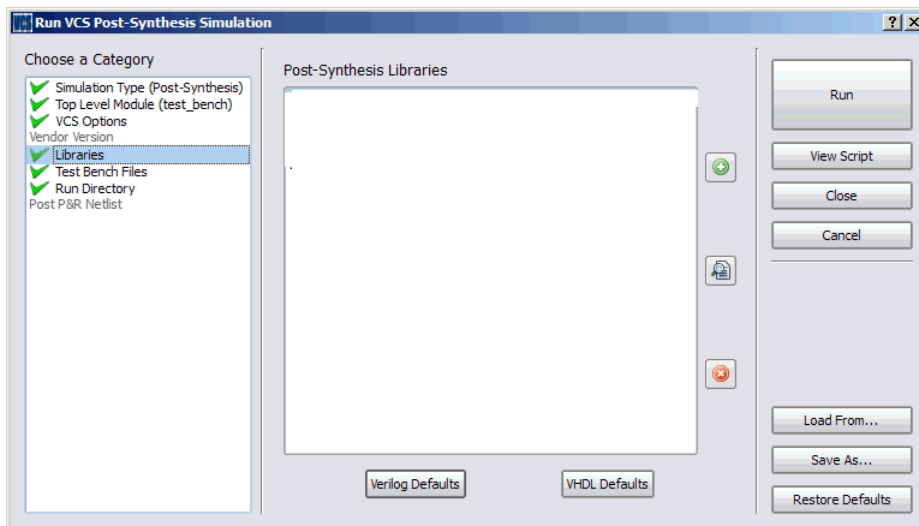
Vendor Version

The following dialog box displays the Vendor Versions task.



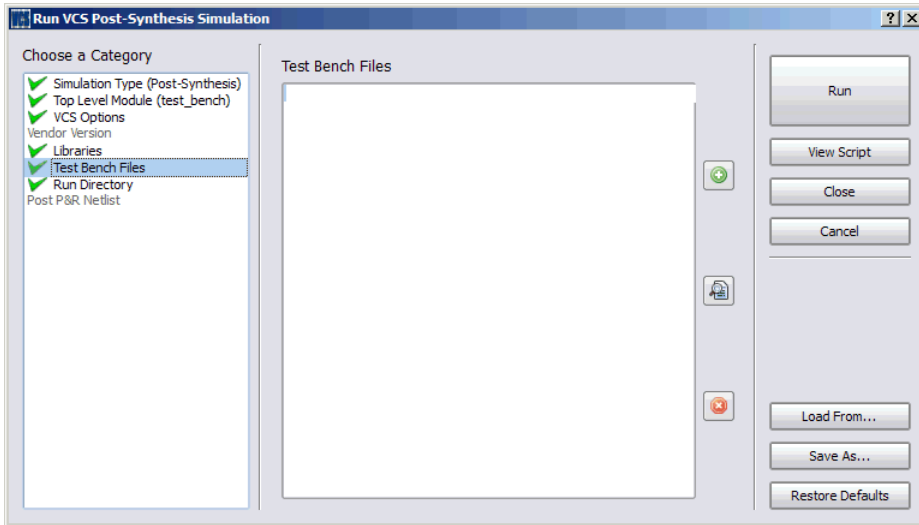
Libraries

The following dialog box displays the Libraries task.



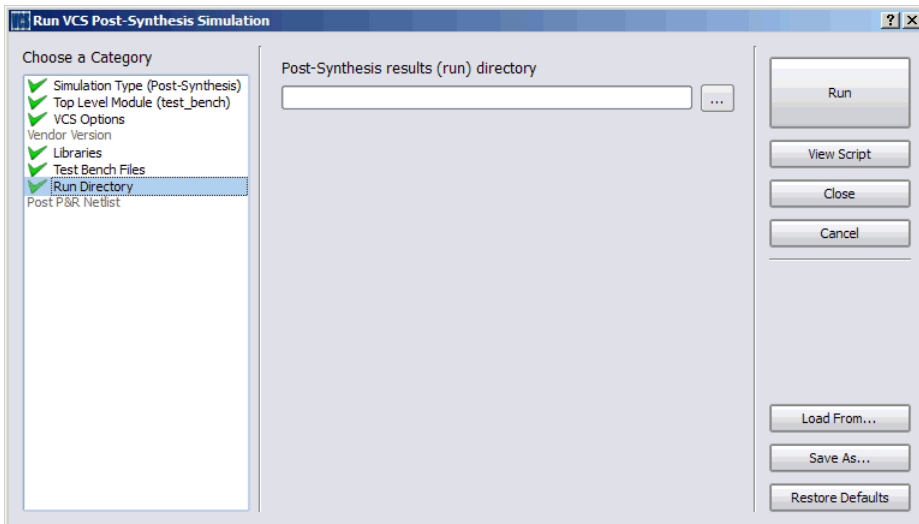
Test Bench Files

The following dialog box displays the Test Bench Files task.



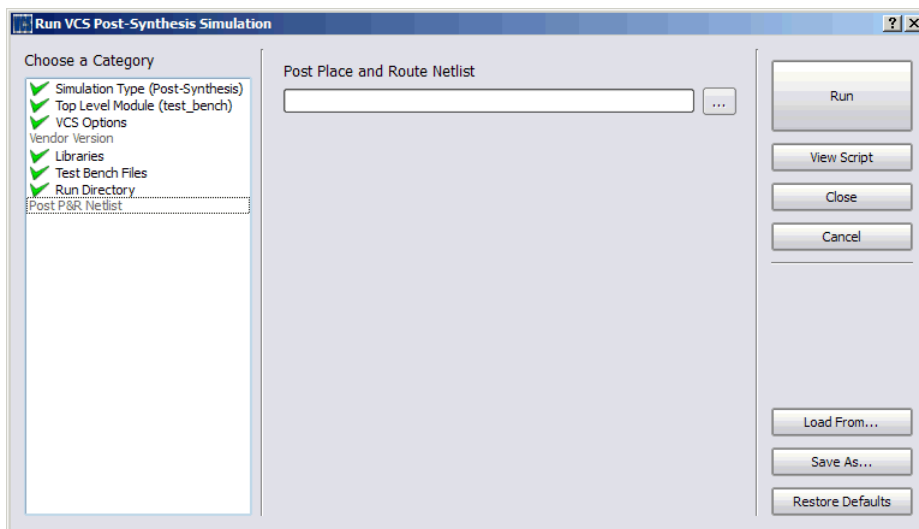
Run Directory

The following dialog box displays the Run Directory task.



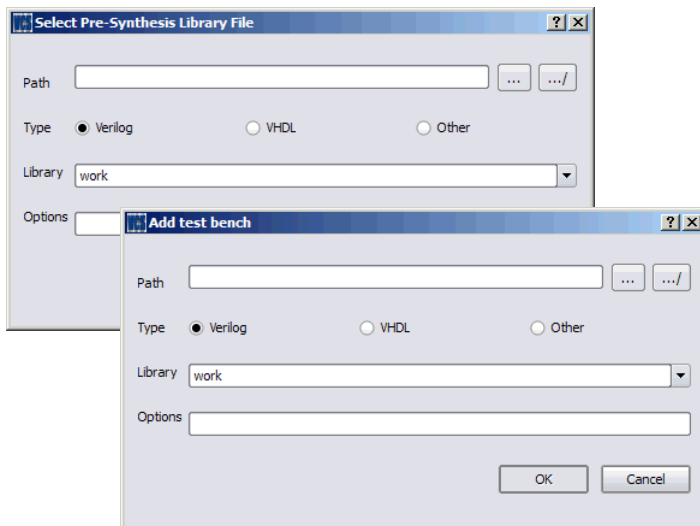
Post P&R Netlist

The following dialog box displays the Post P&R Netlist task.

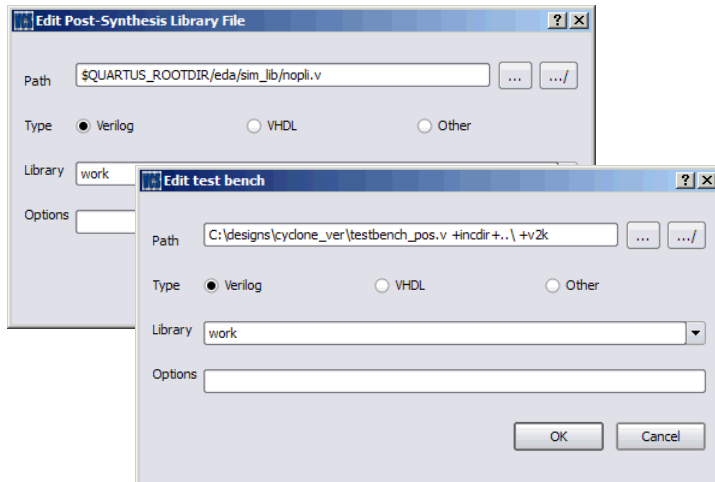


Changing Library and Test Bench Files

You can add Post-synthesis or Post place-and-route library files and test bench files before you launch the VCS simulator. For example, specify options in the following dialog box.



You can also edit library files and test bench files before you launch the VCS simulator. For example: specify options in the following dialog box.



VCS Script File

When you select the VCS Script button on the Run VCS Simulator dialog box, you can view the VCS script generated by the synthesis software for this VCS run. You can also save this VCS script to a file by clicking on **Save a Copy**.

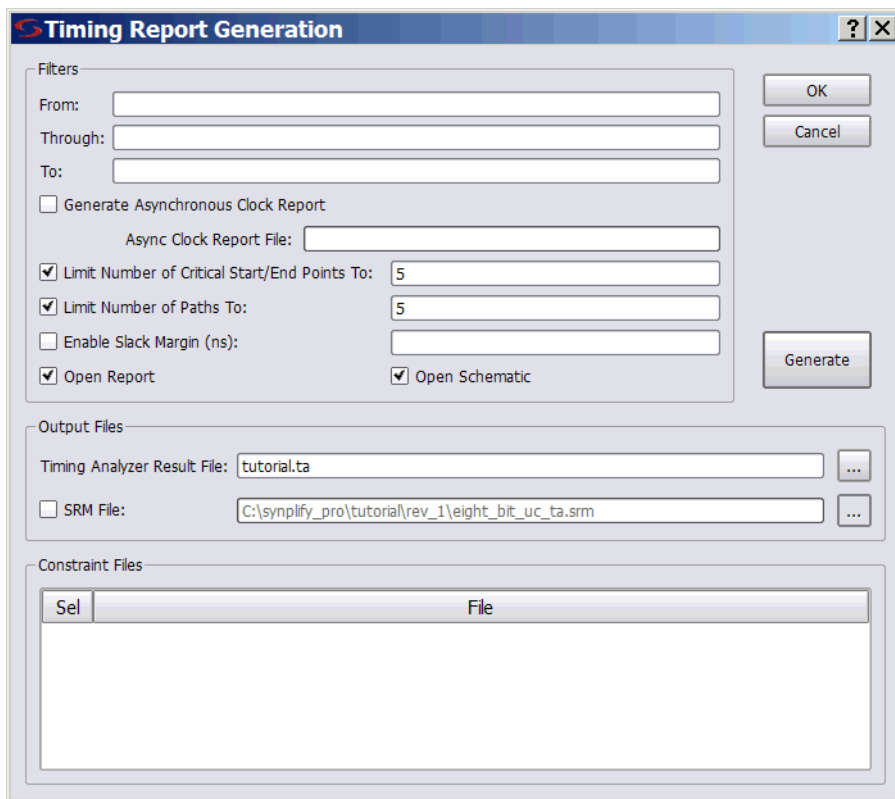
Analysis Menu

When you synthesize a design, a default timing report is automatically written to the log file (*projectName.srr*), located in the results directory. This report provides a clock summary, I/O timing summary, and detailed critical path information for the design. However, you can also generate a custom timing report that provides more information than the default report (specific paths or more than five paths) or one that provides timing based on additional analysis constraint files without rerunning synthesis.

Command	Description
Timing Analyst	<p>Displays the Timing Report Generation dialog box to specify parameters for a stand-alone customized report. See Timing Report Generation Parameters, on page 292 for information on setting these options, and Analyzing Timing in Schematic Views, on page 324 in the <i>User Guide</i> for more information.</p> <p>If you click OK in the dialog box, the specified parameters are saved to a file. To run the report, click Generate. The report is created using your specified parameters.</p>
Generate Timing	<p>Generates and displays a report using the timing option parameters specified above. See the following:</p> <ul style="list-style-type: none"> • Generating Custom Timing Reports with STA, on page 332 for specifics on how to run this report. • Timing Report Generation Parameters, on page 292 for information on setting parameters for the report. This includes information on filtering and options for running backannotation data and power consumption reports.

Timing Report Generation Parameters

You can use the Analysis->Timing Analyst command to specify parameters for a stand-alone timing report. See [Timing Reports, on page 259](#) for information on the file contents.



The **Timing Report Generation** dialog box is used to configure parameters for a timing report. It includes sections for filters, output files, and constraint files.

Filters

- From:
- Through:
- To:
- ☐ Generate Asynchronous Clock Report
 - Async Clock Report File:
- ☒ Limit Number of Critical Start/End Points To:
- ☒ Limit Number of Paths To:
- ☐ Enable Slack Margin (ns):
- ☒ Open Report ☒ Open Schematic

Output Files

- Timing Analyzer Result File:
- ☐ SRM File:

Constraint Files

Sel	File
-----	------

Buttons: OK, Cancel, Generate

The following table provides brief descriptions of the parameters for running a stand-alone timing report.

Timing Report 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 Combining Path Filters for the Timing Analyzer, on page 296 for examples of using filters.</p> <p>Tcl equivalent: set_option -reporting_filter "-from {object1} -to {object2}"</p>
Through	<p>Reports all paths through the specified point or list of objects. See for more information on using this filter. 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"> • Timing Analyzer Through Points, on page 295 • Combining Path Filters for the Timing Analyzer, on page 296. <p>Tcl equivalent: set_option -reporting_filter "-from {object1} -to {object2} -through {object3}"</p>
Generate Asynchronous Clock Report	<p>Generates a report for paths that cross between clock groups. Generally paths in different clock groups are automatically handled as false paths. This option provides a file that contains information on each of the paths and can be viewed in a spreadsheet. This file is in the results directory (<i>projectName_async_clk.rpt.csv</i>). For details on the report, see Asynchronous Clock Report, on page 266.</p> <p>Tcl equivalent: set_option -reporting_async_clock 0 1</p>
Limit Number of Critical Start/End Points to	<p>Specifies the maximum number of start/end paths to display for critical paths in the design. The default is 5. Use this option in combination with the others in the Filters section of the dialog box.</p> <p>Tcl equivalent: set_option -num_startend_points numberOfPaths</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.</p> <p>Tcl equivalent: set_option -reporting_number_paths numberOfPaths</p>

Timing Report Option	Description
Enable Slack Margin (ns)	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. Tcl equivalent: set_option -reporting_margin slackValue
Open Report	When enabled, clicking the Generate button opens the Text Editor on the generated custom timing report specified in the timing report file (ta).
Open Schematic	When enabled, clicking the Generate button opens a Technology view showing the netlist specified in the timing report netlist file (srm). Tcl equivalent: set_option -reporting_output_srm 0 1
Output Files	Displays the name of the generated report: <ul style="list-style-type: none"> • Async Clock Report File contains the spreadsheet data for the asynchronous clock report. This file is not automatically opened when report generation is complete. You can locate this file in the results directory. Default name is <i>projectName_async_clk.rpt.csv</i> (name cannot be changed). Tcl equivalent: set_option -reporting_async_clock 0 1 • Timing Analyst Results File is the standard timing report file, located in the Implementation Results directory. The file is also listed in the Project view. Default filename is <i>projectName.ta</i>. Tcl equivalent: set_option -reporting_filename filename.ta • SRM File updates the Technology view so that you can display the results of the timing updates in the HDL and Physical Analyst tools. The file is also listed in the Project view. Tcl equivalent: set_option -reporting_netlist filename For more details on any of these reports, see Timing Reports, on page 259 .
Constraint Files	Enables analysis design constraint files (adc) to be used for stand-alone timing analysis only. See Input Files, on page 242 for information on this file.
Generate	Clicking this button generates the specified timing report file and timing view netlist file (srm) if requested, saves the current dialog box entries for subsequent use, then closes the dialog box.

Timing Analyzer Through Points

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

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 [Filtering Points: OR List of Through Points, on page 295](#).

AND list Enter the points in a product of sums (POS) format. The tool treats them as an AND list, and only reports the path if it passes through all the points in the list. The POS format for the timing report is the same as for timing constraints. The POS format is as follows:

$$\{n:b \ n:c\}, \{n:d \ n:e\}$$

This constraint translates as follows:

```
b AND d
OR b AND e
OR c AND d
OR c AND e
```

See [Filtering Points: AND List of Through Points, on page 296](#).

See [Defining From/To/Through Points for Timing Exceptions, on page 132](#) in the *User Guide* for more information about specifying through points.

Filtering Points: OR List of Through Points

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.)

The image shows two overlapping screenshots of the 'Filters' dialog box. The left screenshot shows the 'Through' field containing the text 'p:bdpol n:alu_cout'. The right screenshot shows the same dialog box with additional settings: 'Limit Number of Critical Start/End Points To:' is set to 5, 'Limit Number of Paths To:' is set to 5, and 'Open Schematic' is checked. The 'Open Report' checkbox is also checked in both screenshots.

Filtering Points: AND List of Through Points

This example reports the five worst paths passing through port bdpol and net aluout. Enclose each list in braces `{ }` and separate the lists with a comma.

The image shows a single screenshot of the 'Filters' dialog box. The 'Through' field contains the text '{p:bdpol},{n:alu_cout}'. The 'Limit Number of Critical Start/End Points To:' and 'Limit Number of Paths To:' are both set to 5. The 'Open Report' and 'Open Schematic' checkboxes are both checked.

Combining Path Filters for the Timing Analyzer

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 RTL or Technology 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 Technology view rather than the RTL view.

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

Filtering Points: Single Register to Single Register

The screenshot shows the 'Filters' dialog box with the following settings:

- From: `i:op_a_reg`
- Through: (empty)
- To: `i:d_out_reg`
- ☒ Limit Number of Critical Start/End Points To: `5`
- ☒ Limit Number of Paths To: `5`
- ☐ Enable Slack Margin (ns): (empty)
- ☒ Open Report
- ☒ Open Schematic

Filtering Points: Clock Object to Single Register

The screenshot shows the 'Filters' dialog box with the following settings:

- From: `c:clk1`
- Through: (empty)
- To: `i:mult_reg`
- ☒ Limit Number of Critical Start/End Points To: `5`
- ☒ Limit Number of Paths To: `5`
- ☐ Enable Slack Margin (ns): (empty)
- ☒ Open Report
- ☒ Open Schematic

Filtering Points: Single Bit of a Bus to Single Register

The screenshot shows the 'Filters' dialog box with the following settings:

- From: `p:op_reg[1]`
- Through: (empty)
- To: `i:mult_reg`
- ☒ Limit Number of Critical Start/End Points To: `5`
- ☒ Limit Number of Paths To: `5`
- ☐ Enable Slack Margin (ns): (empty)
- ☒ Open Report
- ☒ Open Schematic

Filtering Points: Single Bit of a Bus to Single Bit of a Bus

The screenshot shows the 'Filters' dialog box with the following settings:

- From: `p:op_reg[4]`
- Through: (empty)
- To: `p:d_out_reg[6]`
- ☒ Limit Number of Critical Start/End Points To: `5`
- ☒ Limit Number of Paths To: `5`
- ☐ Enable Slack Margin (ns): (empty)
- ☒ Open Report
- ☒ Open Schematic

Filtering Points: Multiple Bits of a Bus to Multiple Bits of a Bus

The screenshot shows the 'Filters' dialog box with the following settings:

- From: `p:op_a_reg[7:0]`
- Through: (empty)
- To: `p:d_out_reg[15:0]`
- ☒ Limit Number of Critical Start/End Points To: `5`
- ☒ Limit Number of Paths To: `5`
- ☐ Enable Slack Margin (ns): (empty)
- ☒ Open Report
- ☒ Open Schematic

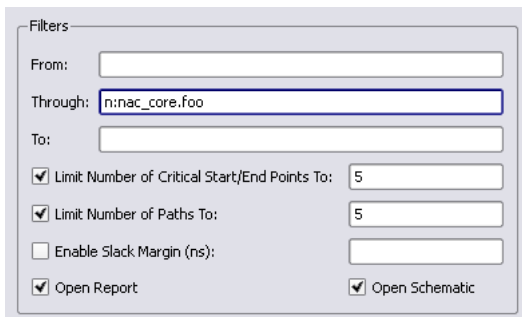
Filtering Points: With Hierarchy

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

The screenshot shows the 'Filters' dialog box with the following settings:

- From: `i:nac_core.rxu_fifo.reg`
- Through: (empty)
- To: `i:nac_core.rxu_channel.reg`
- ☒ Limit Number of Critical Start/End Points To: `5`
- ☒ Limit Number of Paths To: `5`
- ☐ Enable Slack Margin (ns): (empty)
- ☒ Open Report
- ☒ Open Schematic

Filtering Points: Through Point for a Net



Filters

From:

Through:

To:

☒ Limit Number of Critical Start/End Points To:

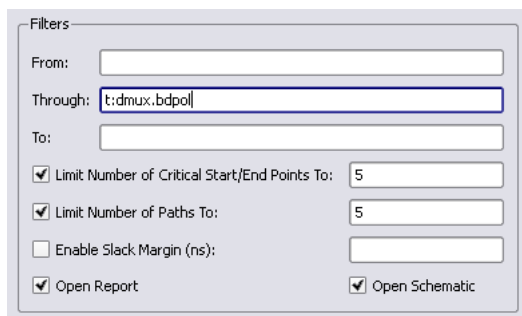
☒ Limit Number of Paths To:

☐ Enable Slack Margin (ns):

☒ Open Report ☒ Open Schematic

Filtering Points: Through Point for a Hierarchical Port

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



Filters

From:

Through:

To:

☒ Limit Number of Critical Start/End Points To:

☒ Limit Number of Paths To:

☐ Enable Slack Margin (ns):

☒ Open Report ☒ Open Schematic

Examples Using 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

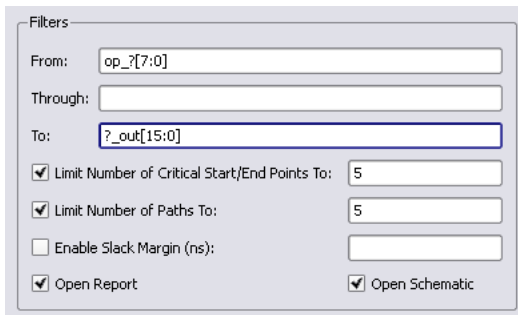
The ? matches single characters. If a design has buses `op_a[7:0]`, `op_b[7:0]`, and `op_c[7:0]`, and you want to filter the paths starting at each of these buses, specify the start points as `op_?[7:0]`. See [Example: ? Wildcard in the Name, on page 301](#) for another example.

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_*[7:0]`. The report shows all paths beginning at each of these buses and for all of the bits of each bus. See [Example: * Wildcard in the Name \(With Hierarchy\), on page 302](#) and [Example: * Wildcard in the Bus Index, on page 302](#) for more examples.

Example: ? Wildcard in the Name

The ? is not supported in bus indices.

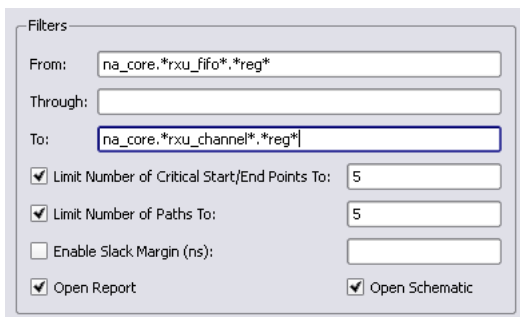


The screenshot shows the 'Filters' dialog box with the following settings:

- From:** `op_?[7:0]`
- Through:** (empty)
- To:** `?_out[15:0]`
- ☒ **Limit Number of Critical Start/End Points To:** `5`
- ☒ **Limit Number of Paths To:** `5`
- ☐ **Enable Slack Margin (ns):** (empty)
- ☒ **Open Report**
- ☒ **Open Schematic**

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.

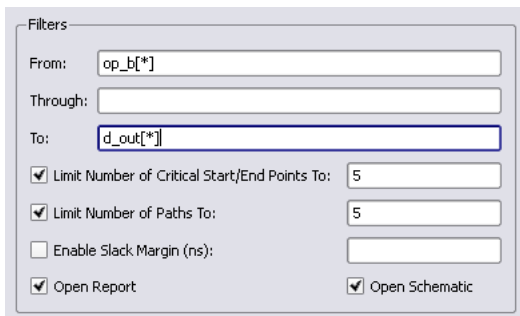


The screenshot shows the 'Filters' dialog box with the following settings:

- From: `na_core.*rxu_fifo*.reg*`
- Through: (empty)
- To: `na_core.*rxu_channel*.reg*`
- ☒ Limit Number of Critical Start/End Points To:
- ☒ Limit Number of Paths To:
- ☐ Enable Slack Margin (ns):
- ☒ Open Report
- ☒ Open Schematic

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.



The screenshot shows the 'Filters' dialog box with the following settings:

- From: `op_b[*]`
- Through: (empty)
- To: `d_out[*]`
- ☒ Limit Number of Critical Start/End Points To:
- ☒ Limit Number of Paths To:
- ☐ Enable Slack Margin (ns):
- ☒ Open Report
- ☒ Open Schematic

HDL Analyst Menu

In the Project View, the HDL Analyst menu contains commands that provide project analysis in the following views:

- [RTL View](#)
- [Technology View](#)



This section describes the HDL Analyst menu commands for the RTL and Technology views. Commands may be disabled (grayed out), depending on the current context. Generally, the commands enabled in any context reflect those available in the corresponding popup menus. The descriptions in the table indicate when commands are context-dependent. For explanations about the terms used in the table, such as filtered and unfiltered, transparent and opaque, see [Filtered and Unfiltered Schematic Views, on page 96](#) and [Transparent and Opaque Display of Hierarchical Instances, on page 101](#). For procedures on using the HDL Analyst tool, see [Analyzing With the HDL Analyst Tool, on page 298](#) of the *User Guide*.

For ease of use, the commands have been divided into sections that correspond to the divisions in the HDL Analyst menu.

- [HDL Analyst Menu: RTL and Technology View Submenus, on page 303](#)
- [HDL Analyst Menu: Hierarchical and Current Level Submenus, on page 304](#)
- [HDL Analyst Menu: Filtering and Flattening Commands, on page 306](#)
- [HDL Analyst Menu: Timing Commands, on page 310](#)
- [HDL Analyst Menu: Analysis Commands, on page 310](#)
- [HDL Analyst Menu: Selection Commands, on page 314](#)
- [HDL Analyst Menu: FSM Commands, on page 314](#)

HDL Analyst Menu: RTL and Technology View Submenus

This table describes the commands that appear on the HDL Analyst->RTL and HDL Analyst->Technology submenus when the RTL or Technology View is active. For procedures on using these commands, see [Analyzing With the HDL Analyst Tool, on page 298](#) of the *User Guide*.

HDL Analyst Command	Description
 RTL->Hierarchical View	Opens a new, hierarchical RTL view. The schematic is unfiltered.
RTL->Flattened View	Opens a new RTL view of your entire design, with a flattened, unfiltered schematic at the level of generic logic cells. See Usage Notes for Flattening, on page 308 for some usage tips.
 Technology->Hierarchical View	Opens a new, hierarchical Technology view. The schematic is unfiltered.
Technology->Flattened View	Creates a new Technology view of your entire design, with a flattened, unfiltered schematic at the level of technology cells. See Usage Notes for Flattening, on page 308 for tips about flattening.
Technology->Flattened to Gates View	Creates a new Technology view of your entire design, with a flattened, unfiltered schematic at the level of Boolean logic gates. See Usage Notes for Flattening, on page 308 for tips about flattening.
Technology->Hierarchical Critical Path	Creates a new Technology view of your design, with a hierarchical, <i>filtered</i> schematic showing only the instances and paths whose slack times are within the slack margin you specified in the Slack Margin dialog. This command automatically enables HDL Analyst->Show Timing Information.
Technology->Flattened Critical Path	Creates a new Technology view of your design, with a flattened, <i>filtered</i> schematic showing only the instances and paths whose slack times are within the slack margin you specified in the Slack Margin dialog. This command automatically enables HDL Analyst->Show Timing Information. See Usage Notes for Flattening, on page 308 for tips about flattening.

HDL Analyst Menu: Hierarchical and Current Level Submenus


This table describes the commands on the HDL Analyst->Hierarchical and HDL Analyst->Current Level submenus. For procedures on using these commands, see [Analyzing With the HDL Analyst Tool, on page 298](#) of the *User Guide*.

HDL Analyst Command	Description
Hierarchical->Expand	<p>Expands paths from selected pins and/or ports up to the nearest objects on any hierarchical level, according to pin/port directions. The result is a <i>filtered</i> schematic. Operates hierarchically, on lower schematic levels as well as the current level.</p> <p>Successive Expand commands expand the paths further, based on the new current selection.</p>
Hierarchical->Expand to Register/Port	<p>Expands paths from selected pins and/or ports, in the port/pin direction, up to the next register, port, or black box. The result is a <i>filtered</i> schematic. Operates hierarchically, on lower schematic levels as well as the current level.</p>
Hierarchical->Expand Paths	<p>Shows all logic, on any hierarchical level, between two or more selected instances, pins, or ports. The result is a <i>filtered</i> schematic. Operates hierarchically, on lower schematic levels as well as the current level.</p>
Hierarchical->Expand Inwards	<p>Expands within the hierarchy of an instance, from the lower-level ports that correspond to the selected pins, to the nearest objects and no further. The result is a <i>filtered</i> schematic. Operates hierarchically, on lower schematic levels as well as the current level.</p>
Hierarchical->Goto Net Driver	<p>Displays the unfiltered schematic sheet that contains the net driver for the selected net. Operates hierarchically, on lower schematic levels as well as the current level.</p>
Hierarchical->Select Net Driver	<p>Selects the driver for the selected net. The result is a <i>filtered</i> schematic. Operates hierarchically, on lower schematic levels as well as the current level.</p>
Hierarchical->Select Net Instances	<p>Selects instances connected to the selected net. The result is a <i>filtered</i> schematic. Operates hierarchically, on lower schematic levels as well as the current level.</p>
Current Level->Expand	<p>Expands paths from selected pins and/or ports up to the nearest objects on the current level, according to pin/port directions. The result is a <i>filtered</i> schematic. Limited to all sheets on the current schematic level. This command is only available if a HDL Analyst view is open.</p> <p>Successive Expand commands expand the paths further, based on the new current selection.</p>

HDL Analyst Command	Description
Current Level->Expand to Register/Port	Expands paths from selected pins and/or ports, according to the pin/port direction, up to the next register, ports, or black box on the current level. The result is a <i>filtered</i> schematic. Limited to all sheets on the current schematic level.
Current Level->Expand Paths	Shows all logic on the current level between two or more selected instances, pins, or ports. The result is a <i>filtered</i> schematic. Limited to the current schematic level (all sheets).
Current Level->Goto Net Driver	Displays the unfiltered schematic sheet that contains the net driver for the selected net. Limited to all sheets on the current schematic level.
Current Level->Select Net Driver	Selects the driver for the selected net. The result is a <i>filtered</i> schematic. Limited to all sheets on the current schematic level.
Current Level->Select Net Instances	Selects instances on the current level that are connected to the selected net. The result is a <i>filtered</i> schematic. Limited to all sheets on the current schematic level.

HDL Analyst Menu: Filtering and Flattening Commands

This table describes the filtering and flattening commands on the HDL Analyst menu. For procedures on filtering and flattening, see [Analyzing With the HDL Analyst Tool, on page 298](#) of the *User Guide*.

HDL Analyst Command	Description
 Filter Schematic	Filters your entire design to show only the selected objects. The result is a <i>filtered</i> schematic. For more information about using this command, see Filtering Schematics, on page 302 of the <i>User Guide</i> . This command is only available with an open HDL Analyst view.

HDL Analyst Command	Description
---------------------	-------------

Flatten Current Schematic (Unfiltered Schematic)	
---	--

	In an unfiltered schematic, the command flattens the current schematic, at the current level and all levels below. In an RTL view, the result is at the generic logic level. In a Technology view, the result is at the technology-cell level. See the next table entry for information about flattening a filtered schematic.
--	--

This command does not do the following:

- Flatten your entire design (unless the current level is the top level)
- Open a new view window
- Take into account the number of Dissolve Levels defined in the Schematic Options dialog box

See [Usage Notes for Flattening, on page 308](#) for tips.

HDL Analyst Command Description

**Flatten Current Schematic
(Filtered Schematic)**

In a filtered schematic, flattening is a two-step process:

- Only unhidden transparent instances (including nested ones) are flattened in place, in the context of the entire design. Opaque and hidden hierarchical instances remain hierarchical. The effect of this command is that all hollow boxes with pale yellow borders are removed from the schematic, leaving only what was displayed inside them.
- The original filtering is restored.

In an RTL view, the result is at the generic logic level. In a Technology view, the result is at the technology-cell level.

This command does not do the following:

- Flatten everything inside a transparent instance. It only flattens transparent instances and any nested transparent instances they contain.
- Open a new view window.
- Take into account the number of Dissolve Levels defined in the Schematic Options dialog box.

See [Usage Notes for Flattening, on page 308](#) for usage tips.

**Unflatten Current
Schematic**

Undoes any flattening operations and returns you to the original schematic, as it was before flattening and any filtering.

This command is available only if you have explicitly flattened a hierarchical schematic using HDL Analyst->Flatten Current Schematic, for example. It is not available for flattened schematics created directly with the RTL and Technology submenus of the HDL Analyst menu.

Usage Notes for Flattening

It is usually more memory-efficient to flatten only parts of your design, as needed. The following are a few tips for flattening designs with different commands. For detailed procedures, see [Flattening Schematic Hierarchy, on page 310](#) of the *User Guide*.

RTL/Technology->Flattened View Commands

- Use Flatten Current Schematic to flatten only the current hierarchical level and below.
- Flatten selected hierarchical instances with Dissolve Instances (followed by Flatten Current Schematic, if the schematic is filtered).
- To make hierarchical instances transparent without flattening them, use Dissolve Instances in a filtered schematic. This shows their details nested inside the instances.

Flatten Current Schematic Command (Unfiltered View)


- Flatten selected hierarchical instances with Dissolve Instances.
- To see the lower-level logic inside a hierarchical instance, push into it instead of flattening.
- Selectively flatten your design by hiding the instances you do not need, flattening, and then unhiding the instances.
- Flattening erases the history of displayed sheets for the current view. You can no longer use View->Back. You can, however, use UnFlatten Schematic to get an unflattened view of the design.

Flatten Current Schematic Command (Filtered View)

- Flatten selected hierarchical instances with Dissolve Instances, followed by Flatten Current Schematic.
 - Selectively flatten your design by hiding the instances you do not need, flattening, and then unhiding the instances.
 - Flattening erases the history of displayed sheets for the current view. You can no longer use View->Back. You can do the following:
 - Use View->Back for a view of the transparent instance flattened in the context of the entire design. This is the view generated after step 1 of the two-step flattening process described above. Use UnFlatten Schematic to get an unflattened view of the design.
-

HDL Analyst Menu: Timing Commands

This table describes the timing commands on the HDL Analyst menu. For procedures on using the timing commands, see [Analyzing With the HDL Analyst Tool, on page 298](#) of the *User Guide*.

HDL Analyst Command	Description
Set Slack Margin	Displays the Slack Margin dialog box, where you set the slack margin. HDL Analyst->Show Critical Path displays only those instances whose slack times are worse than the limit set here. Available only in a Technology view.
 Show Critical Path	Filters your entire design to show only the instances and paths whose slack times exceed the slack margin set with Set Slack Margin, above. The result is flat if the entire design was already flat. This command also enables Show Timing Information (see below). Available only in a Technology view.
Show Timing Information	When enabled, Technology view schematics are annotated with timing numbers above each instance. The first number is the cumulative path delay; the second is the slack time of the worst path through the instance. Negative slack indicates that timing has not met requirements. Available only in a Technology view. For more information, see Viewing Timing Information, on page 324 on the <i>User Guide</i> .

HDL Analyst Menu: Analysis Commands

This table describes the analysis commands on the HDL Analyst menu. For procedures on using the analysis commands, see [Analyzing With the HDL Analyst Tool, on page 298](#) of the *User Guide*.

HDL Analyst Command	Description
Isolate Paths	<p>Filters the current schematic to display only paths associated with all the pins of the selected instances. The paths follow the pin direction (from output to input pins), up to the next register, black box, port, or hierarchical instance.</p> <p>If the selected objects include ports and/or pins on unselected instances, the result also includes paths associated with those selected objects.</p> <p>The range of the operation is all sheets of a filtered schematic or just the current sheet of an unfiltered schematic. The result is always a filtered schematic.</p> <p>In contrast to the Expand operations, which add to what you see, Isolate Paths can only remove objects from the display. While Isolate Paths is similar to Expand to Register/Port, Isolate Paths reduces the display while Expand to Register/Port augments it.</p>
Show Context	<p>Shows the original, unfiltered schematic sheet that contains the selected instance. Available only in a filtered schematic.</p>
Hide Instances	<p>Hides the logic inside the selected hierarchical (non-primitive) instances. This affects only the active HDL Analyst view; the instances are not hidden in other HDL Analyst views.</p> <p>The logic inside hidden instances is not loaded (saving dynamic memory), and it is unrecognized by searching, dissolving, flattening, expansion, and push/pop operations. (Crossprobing does recognize logic inside hidden instances, however.) See Usage Notes for Hiding Instances, on page 313 for tips.</p>
Unhide Instances	<p>Undoes the effect of Hide Instances: the selected hidden hierarchical instances become visible (susceptible to loading, searching, dissolving, flattening, expansion, and push/pop operations). This affects only the current HDL Analyst view; the instances are not hidden in other HDL Analyst views.</p>

HDL Analyst Command	Description
Show All Hier Pins	Shows all pins on the selected transparent, non-primitive instances. Available only in a filtered schematic. Normally, transparent instance pins that are connected to logic that has been filtered out are not displayed. This command lets you display these pins that connected to logic that has been filtered out. Pins on primitives are always shown.
Dissolve Instances	Shows the lower-level details of the selected non-hidden hierarchical instances. The number of levels dissolved is determined by the Dissolve Levels value in the HDL Analyst Options dialog box (New HDL Analyst Options Command, on page 329). For usage tips, see Usage Notes for Dissolving Instances, on page 313 .
Dissolve to Gates	<p>Dissolves the selected instances by flattening them to the gate level. This command displays the lower-level hierarchy of selected instances, but it dissolves technology primitives as well as hierarchical instances. Technology primitives are dissolved to generic synthesis symbols. The command is only available in the Technology view.</p> <p>The number of levels dissolved is determined by the Dissolve Levels value in the HDL Analyst Options dialog box (New HDL Analyst Options Command, on page 329).</p> <p>Dissolving an instance one level redraws the current sheet, replacing the hierarchical dissolved instance with the logic you would see if you pushed into it using Push/pop mode. Unselected objects or selected hidden instances are not dissolved.</p> <p>The effect of the command varies:</p> <ul style="list-style-type: none"> • In an unfiltered schematic, this command <i>flattens</i> the selected instances. This means the history of displayed sheets is removed. The resulting schematic is unfiltered. • In a filtered schematic, this command makes the selected instances <i>transparent</i>, displaying their internal, lower-level logic inside hollow boxes. History is retained. You can use Flatten Schematic to flatten the transparent instances, if necessary. The resulting schematic if filtered.

Usage Notes for Hiding Instances

The following are a few tips for hiding instances. For detailed procedures, see [Flattening Schematic Hierarchy, on page 310](#) of the *User Guide*.

- Hiding hierarchical instances soon after startup can often save memory. After the interior of an instance has been examined (by searching or displaying), it is too late for this savings.
- You can save memory by creating small, temporary working files: File->Save As .srs or .srm files does not save the hidden logic (hidden instances are saved as black boxes). Restarting the synthesis tool and loading such a saved file can often result in significant memory savings.
- You can selectively flatten instances by temporarily hiding all the others, flattening, then unhiding.
- You can limit the range of Edit->Find (see [Find Command \(HDL Analyst\), on page 183](#)) to prevent it looking inside given instances, by temporarily hiding them.

Usage Notes for Dissolving Instances

Dissolving an instance one level redraws the current sheet, replacing the hierarchical dissolved instance with the logic you would see if you pushed into it using Push/pop mode. Unselected objects or selected hidden instances are not dissolved. For additional information about dissolving instances, see [Flattening Schematic Hierarchy, on page 310](#) of the *User Guide*.

The type (filtered or unfiltered) of the resulting schematic is unchanged from that of the current schematic. However, the effect of the command is different in filtered and unfiltered schematics:

- In an unfiltered schematic, this command flattens the selected instances. This means the history of displayed sheets is removed.
- In a filtered schematic, this command makes the selected instances transparent, displaying their internal, lower-level logic inside hollow boxes. History is retained. You can use Flatten Schematic to flatten the transparent instances, if necessary. This command is only available if an HDL Analyst view is open.

HDL Analyst Menu: Selection Commands

This table describes the selection commands on the HDL Analyst menu.

HDL Analyst Command	Description
Select All Schematic ->Instances ->Ports	Selects all Instances or Ports, respectively, on all sheets of the current schematic. All other objects are unselected. This does not select objects on other schematics.
Select All Sheet ->Instances ->Ports	Selects all Instances or Ports, respectively, on the current schematic sheet. All other objects are unselected.
Unselect All	Unselects all objects in all HDL Analyst views.

HDL Analyst Menu: FSM Commands

This table describes the FSM commands on the HDL Analyst menu.

HDL Analyst Command	Description
View FSM	Displays the selected finite state machine in the FSM Viewer. Available only in an RTL view.
View FSM Info File	Displays information about the selected finite state machine module, including the number of states, the number of inputs, and a table of the states and transitions. Available only in an RTL view.

Options Menu

Use the Options menu to configure the VHDL and Verilog compilers, customize toolbars, and set options for the Project view, Text Editor, and HDL Analyst schematics. When using certain technologies, additional menu commands let you run technology-vendor software from this menu.

The following table describes the Options menu commands.

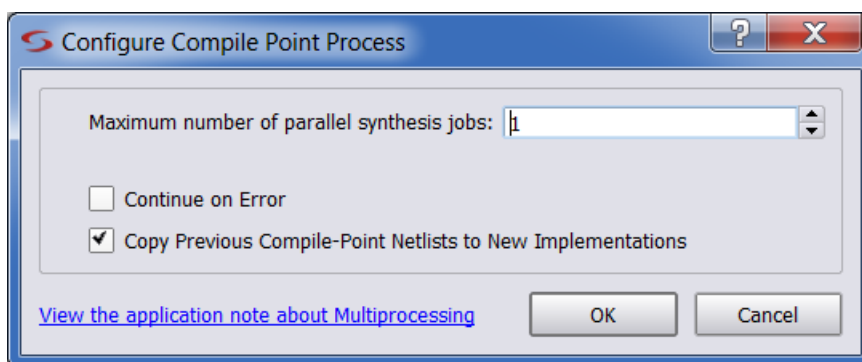
Command	Description
Basic Options Menu Commands for all Views	
Configure VHDL Compiler	Opens the Implementation Options dialog box where you can set the top-level entity and the encoding method for enumerated types. State-machine encoding is automatically determined by the FSM compiler or FSM explorer, or you can specify it explicitly using the <code>syn_encoding</code> attribute. See Implementation Options Command , on page 214.
Configure Verilog Compiler	Opens the Implementation Options dialog box where you can specify the top-level module and the include search path. See Implementation Options Command , on page 214.
Configure Compile Point Process	Lets you specify the maximum number of parallel synthesis jobs that can be run and how errors in compile points are treated. See Configure Compile Point Process Command , on page 316.
Toolbars	Lets you customize the toolbars.
Project View Options	Sets options for organizing files in the Project view. See Project View Options Command , on page 319.
Editor Options	Sets your Text Editor syntax coloring, font, and tabs. See Editor Options Command , on page 324.
P&R Environment Options	Displays the environmental variable options set for the place-and-route tool. See Place and Route Environment Options Command , on page 327.
Project Status Page Location	Saves the current project status to a location of your choice. See Project Status Page Location , on page 327.

Command	Description
HDL Analyst Options	Sets display preferences for HDL Analyst schematics (RTL and Technology views). See New HDL Analyst Options Command, on page 329 .
Configure External Programs	Lets you set browser and Acrobat Reader options on Linux platforms. See Configure External Programs Command, on page 339 .
Options Menu Commands Specifically for the Project View	
Configure Identify Launch	If Identify software is not properly installed, you might run into problems when you try to launch it from the synthesis tools. Use the Configure Identify Launch dialog box to help you resolve these issues. For guidelines to follow, see Handling Problems with Launching Identify, on page 556 in the <i>User Guide</i> .

Configure Compile Point Process Command

Use the Configure Compile Point Process command to run multiprocessing with compile points. This option allows the synthesis software to run multiple, independent compile point jobs simultaneously, providing additional runtime improvements for the compile point synthesis flow.

This feature is supported on Windows and Linux for certain technologies only. This command is disabled out for technologies that are not supported.



Field/Option	Description
Maximum Number of Parallel Synthesis Jobs	<p>Sets the maximum number of synthesis jobs that can run in parallel. It displays the current value from the <code>ini</code> file, and allows you to reset it. Use this option for multiprocessing by running compile point jobs in parallel.</p> <p>Set a value based on the number of available licenses. Note that four licenses are used for each job by default. See License Utilization for Multiprocessing, on page 318.</p> <p>When you set this option, it resets the <code>MaxParallelJobs</code> value in the <code>.ini</code> file. See Maximum Parallel Jobs, on page 318 for other ways to specify this value.</p>
Continue on Error	<p>Allows the software to continue on an error in a compile point and synthesize the rest of the design, even when there might be problems with a portion of the design.</p> <p>The Continue on Error mode automatically enables the <code>MultiProcessing</code> option to run with compile points using a single license; this is the default. For additional runtime improvements, you can specify multiple synthesis jobs that run in parallel. See Chapter 14, Improving Runtime for details.</p> <p>For more information about Continue on Error mode during compile-point synthesis, see Combining Compile Points with Multiprocessing, on page 445 in the <i>User Guide</i>.</p> <p>Tcl equivalent: <code>set_option -continue_on_error 0 1</code></p>
Copy Previous Compile-Point Netlists to New Implementation	<p>Allows you to copy compile point netlist (<code>srd</code>) files generated from the previous implementation into a new implementation that has been created.</p>

Maximum Parallel Jobs

There are three ways to specify the maximum number of parallel jobs:

ini File	<p>Set this variable in the MaxParallelJobs variable in the product ini file:</p> <pre>[JobSetting] MaxParallelJobs=<n></pre> <p>This value is used by the UI as well as in batch mode, and is effective until you specify a new value. You can change it with the Options->Configure Compile Point Process command.</p>
Tcl Variable	<p>Set the following variable in a Tcl file, the project files, or from the Tcl window:</p> <pre>set_option -max_parallel_jobs=<n></pre> <p>This is a global option that is applied to all project files and their implementations. This value takes effect immediately. If you set it in the Tcl file or project file, it remains in effect until you specify a new value. If you set it from the Tcl window, the max_parallel_jobs value is only effective for the session and will be lost when you exit the application.</p>
Configure Compile Point Process Command	<p>The Maximum Number of Parallel Synthesis Jobs option displays the current ini file value and allows you to reset it.</p>

License Utilization for Multiprocessing

When you decide to run parallel synthesis jobs, additional licenses may be required for the compile point jobs. By default, four parallel jobs use one license. For example, if you set the Maximum number of parallel synthesis jobs to 12, the synthesis tool consumes one license to run 4 compile point jobs and can utilize the two additional licenses to run 8 more parallel jobs if they are available for your computing environment. Licenses are released as jobs complete, and then consumed by new jobs which need to run.

The actual number of licenses utilized depends on the following:

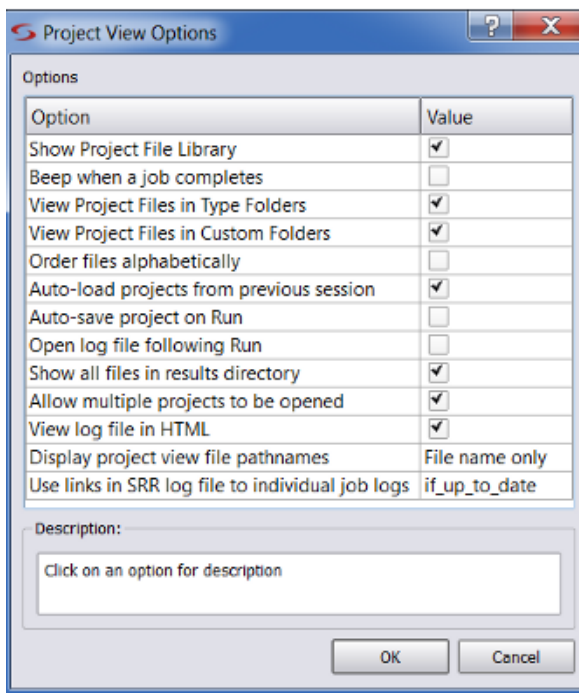
1. Synthesis software scheme for the compile point requirements used to determine the maximum number of parallel jobs or licenses a particular design tries to use.
2. Value set on the Configure Compile Point Process dialog box.

3. Number of licenses actually available. You can use Help->Preferred License Selection to check the number of available license. If you need to increase the number of available licenses, you can specify multiple license types. For more information, see [Using Different License Types for Multiprocessing, on page 536](#).

Note that factors 1 and 3 above can change during a single synthesis run. The number of jobs equals the number of licenses, which then equates the lowest value of these three factors.

Project View Options Command

Select Options->Project View Options to display the Project View Options dialog box, where you define how projects appear and are organized in the Project view.

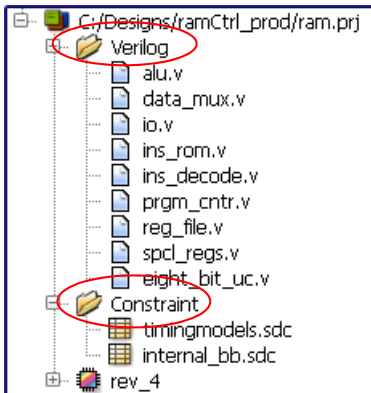


The following table describes the Project View Options dialog box features.

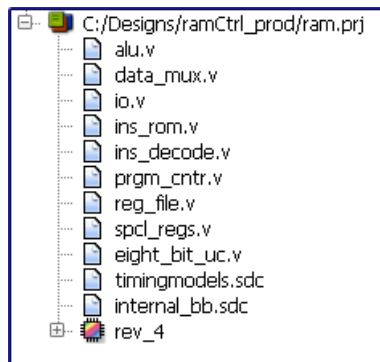
Field/Option	Description
Show Project File Library	When enabled, displays the corresponding VHDL library next to each source VHDL filename, in the Project Tree view of the Project view. For example, with library dune, file pc.vhd is listed as [dune] pc.vhd if this option is enabled, and as pc.vhd if it is disabled. See also Set VHDL Library Command, on page 204 , for how to change the library of a file.
Beep when a job completes	When enabled, sounds an audible signal whenever a project finishes running.
View Project Files in Type Folders	When enabled, organizes project files into separate folders by type. See View Project Files in Type Folders Option, on page 321 and add_file, on page 17 .
View Project Files in Custom Folders	When enabled, allows you to view files contained within the custom folders created for the project. See View Project Files in Custom Folders Option, on page 322 .
Order files alphabetically	When enabled, the software orders the files within folders alphabetically instead of in project order. You can also use the Sort Files option in the Project view.
Autoload projects from previous session	Enable/Disable automatically loading projects from the previous session. Otherwise, projects will not be loaded automatically. This option is enabled by default. See Loading Projects With the Run Command, on page 322 .
Auto-save project on Run	Enable/Disable automatically saving projects when the Run button is selected. See Automatically Save Project on Run, on page 323 .
Open Log file following Run	Enable/Disable automatically opening and displaying log file after a synthesis run.
Show all files in results directory	When enabled, shows all files in the Implementation Results view. When disabled, the results directory shows only files generated by the synthesis tool itself.

Field/Option	Description
Allow multiple projects to be opened	When enabled, multiple projects are displayed at the same time. See Allow Multiple Projects to be Opened Option , on page 322.
View log file in HTML	Enable/Disable viewing of log file report in HTML format versus text format. See Log File , on page 254.
Project file name display	From the drop-down menu, select one the following ways to display project files: <ul style="list-style-type: none"> • File name only • Relative file path • Full file path
Use links in SRR log file to individual job logs	Determines if individual job logs use links in the srr log file. You can select: <ul style="list-style-type: none"> • off—appends individual job logs to the srr log file. • on—always link to individual job logs. • if_up_to_date—only links to individual job logs if the module is up-to-date.

View Project Files in Type Folders Option



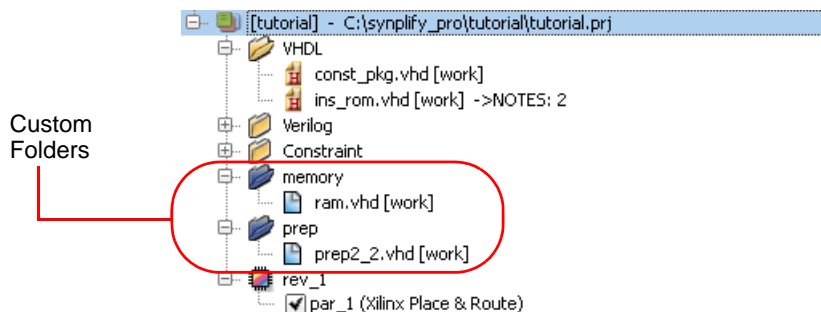
View project files in type folders *enabled*



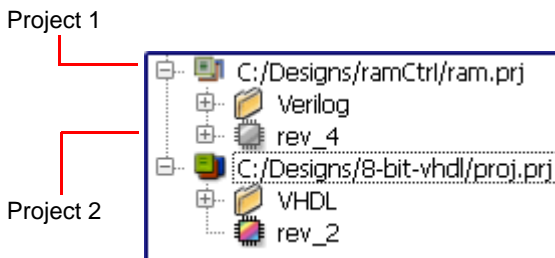
View project files in type folders *disabled*

View Project Files in Custom Folders Option

Selecting this option enables you to view user-defined custom folders that contain a predefined subset of project files in various hierarchy groupings or organizational structures. Custom folders are distinguished by their blue color. For information on creating custom folders, see [Creating Custom Folders](#), on page 64 in the *User Guide*.



Allow Multiple Projects to be Opened Option



Loading Projects With the Run Command

When you load a project that includes the project-run command, a dialog box appears in the Project view with the following message:

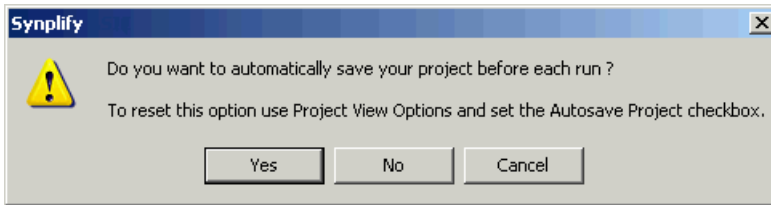
Project run command encountered during project load. Are you sure you want to run?

You can reply with either yes or no.

Automatically Save Project on Run

If you have modified your project on the disk directory since being loaded into the Project view and you run your design, a message is generated that infers the UI is out-of-date.

The following dialog box appears with a message to which you must reply.

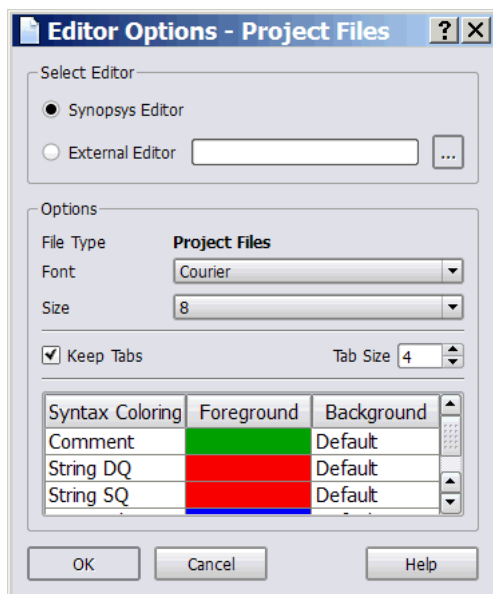


You can specify one of the following:

- Yes — The Auto-save project on Run switch on the Project View Options dialog box is automatically enabled, and then your design is run.
- No — The Auto-save project on Run switch on the Project View Options dialog box is not enabled, but your design is run.
- Cancel — Closes this message dialog box and does not run your design.

Editor Options Command

Select Options->Editor Options to display the Editor Options dialog box, where you select either the internal text editor or an external text editor.



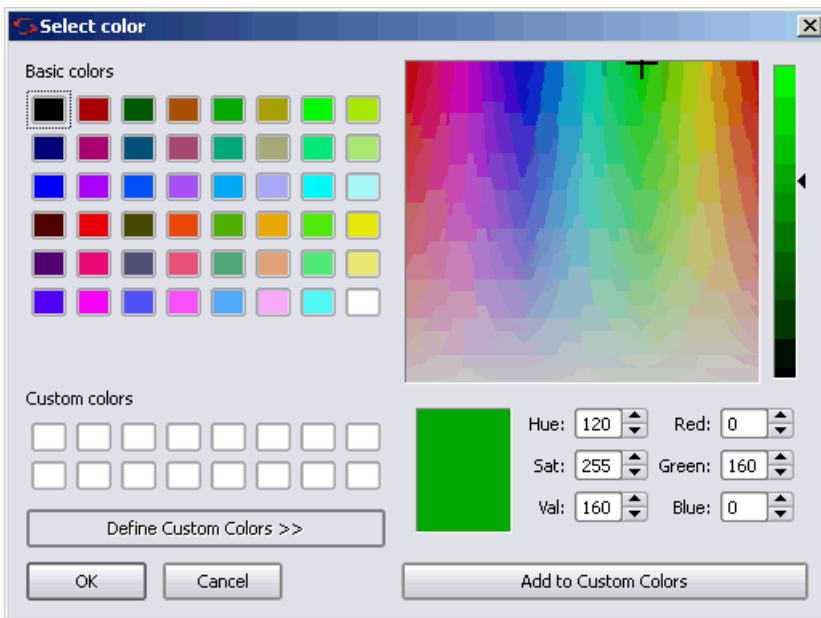
The following table describes the Editor Options dialog box features.

Feature	Description
Select Editor	Select an internal or external editor.
<ul style="list-style-type: none"> Synopsys Editor 	Sets the Synopsys text editor as the default text editor.
<ul style="list-style-type: none"> External Editor 	<p>Uses the specified external text editor program to view text files from within the Synopsys tool. The executable specified must open its own window for text editing. See Using an External Text Editor, on page 39 of the <i>User Guide</i> for a procedure.</p> <p><i>Note:</i> Files opened with an external editor cannot be crossprobed.</p>
Options	Set text editing preferences.
<ul style="list-style-type: none"> File Type 	You can define text editor preferences for the following file types: project files, HDL files, log files, constraint files, and default files.

Feature	Description
• Font	Lets you define fonts to use with the text editor.
• Font Size	Lets you define font size to use with the text editor.
• Keep Tabs • Tab Size	Lets you define whether to use tab settings with the text editor.
• Syntax Coloring	Lets you define foreground or background syntax coloring to use with the text editor. See Color Options, on page 325 .

Color Options

Click in the Foreground or Background field for the corresponding object in the Syntax Coloring field to display the color palette.

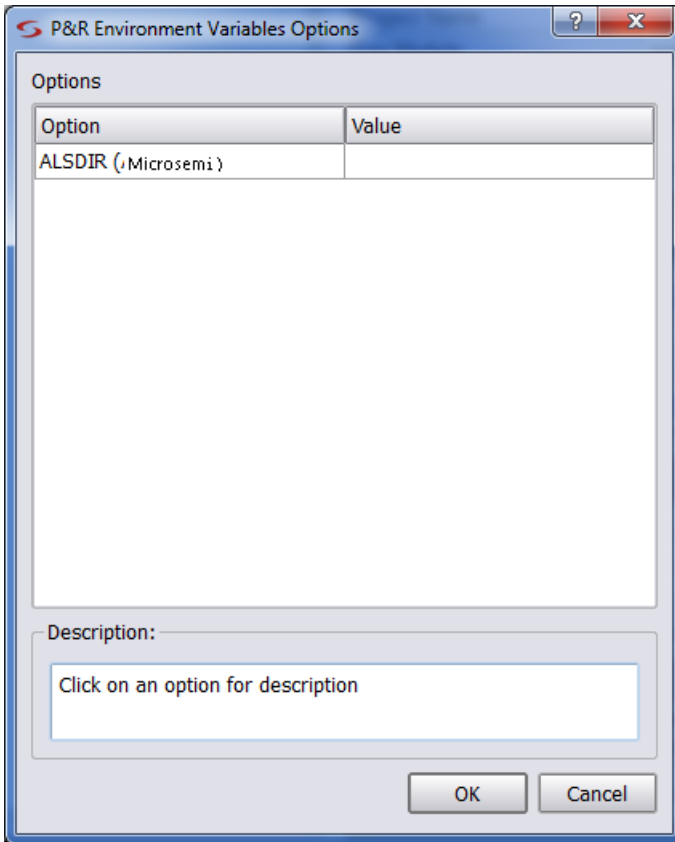


You can set syntax colors for some common syntax options listed in the following table.

Syntax	Description
Comment	Comment strings contained in all file types.
Error	Error messages contained in the log file.
Gates	Gates contained in HDL source files.
Info	Informational messages contained in the log file.
Keywords	Generic keywords contained in the project, HDL source, constraint, and log files.
Line Comment	Line comments contained in the HDL source, C, C++, and log files.
Note	Notes contained in the log file.
SDCKeyword	Constraint-specific keywords contained in the .sdc file.
Strength	Strength values contained in HDL source files.
String DQ	String values within double quotes contained in the project, HDL source, constraint, C, C++, and log files.
String SQ	String values within single quotes contained in the project, HDL source, constraint, C, C++, and log files.
SVKeyword	SystemVerilog keywords contained in the Verilog file.
Types	Type values contained in HDL source files.
Warning	Warning messages contained in the log file.

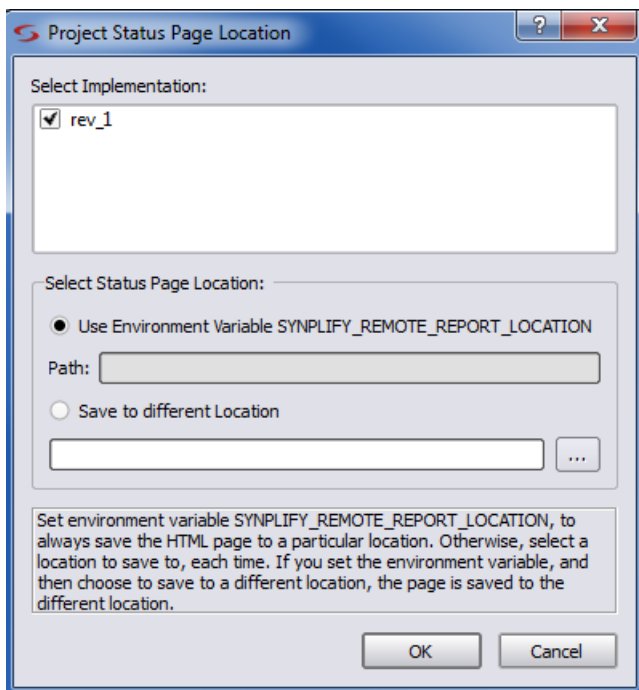
Place and Route Environment Options Command

Select Options->P&R Environment Options to display the environment variable options set for the place-and-route tool. This option allows you to change the specified location of the selected place-and-route tool set on your system; the software locates and runs this updated version of the P&R tool for the current session of the synthesis tool.



Project Status Page Location

Lets you save the current project status to a location of your choice. You can then view the project status offline with any browser on a mobile device.

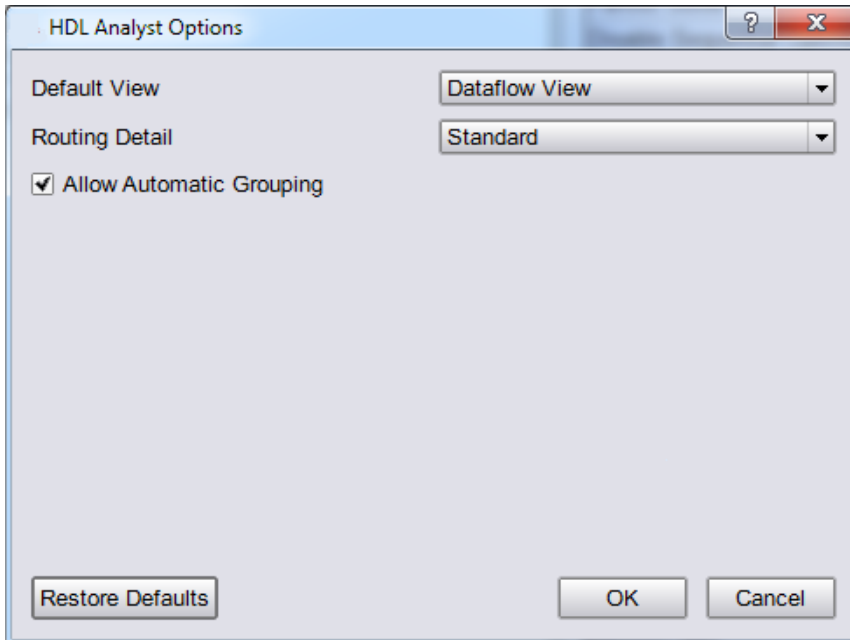


The following table describes the Project Status Page Location dialog box options.

Option	Description
Select Implementation	Select the implementation for the design for which you want synthesis results. You can select multiple implementations.
Select Status Page Location <ul style="list-style-type: none"> • Use Environment Variable SYNPLIFY_REMOTE_REPORT_LOCATION • Save to Different Location 	Select the location on your computer where you want to save the project status reports: <ul style="list-style-type: none"> • Use the environment variable to specify a standard location for the project status reports. Choose this option if you always want to save the reports to the same location. • Choose a location for the project status reports for the current implementation. You can change this as often as you like. For more information, see Accessing Results Remotely, on page 190 in the <i>User Guide</i> .

New HDL Analyst Options Command

Select Options->HDL Analyst Options to display a dialog box where you define preferences for the New HDL Analyst schematic. For details, see [Setting Schematic Preferences](#), on page 338 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"> • Clocks View - Displays all sequential elements connected to clock nets so that clocks in the design can be debugged. • Dataflow View - Displays objects from a left to right datapath flow. This is the default.
Routing Detail	Specify how the tool determines the detailed routing for the design: <ul style="list-style-type: none"> • Optimized • Detailed • Standard - This is the default. • Quick - Direct connections
Allow Automatic Grouping	When enabled, automatic grouping is performed.
Restore Defaults	Click this button to reset all options to their defaults.

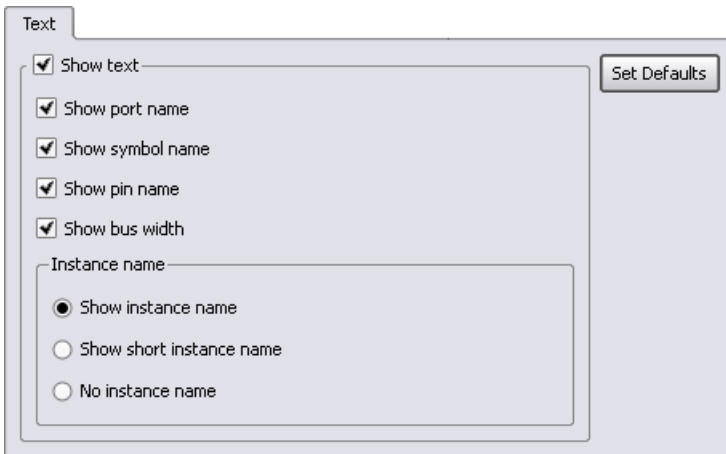
HDL Analyst Options Command

Select Options->HDL Analyst Options to display the HDL Analyst Options dialog box, where you define preferences for the HDL Analyst schematic views (RTL and Technology views). Some preferences take effect immediately, others only take effect in the next view that you open. For details, see [Setting Schematic Preferences, on page 400](#) 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 331](#)
- [General Panel, on page 332](#)
- [Sheet Size Panel, on page 336](#)
- [Visual Properties Panel, on page 338](#)

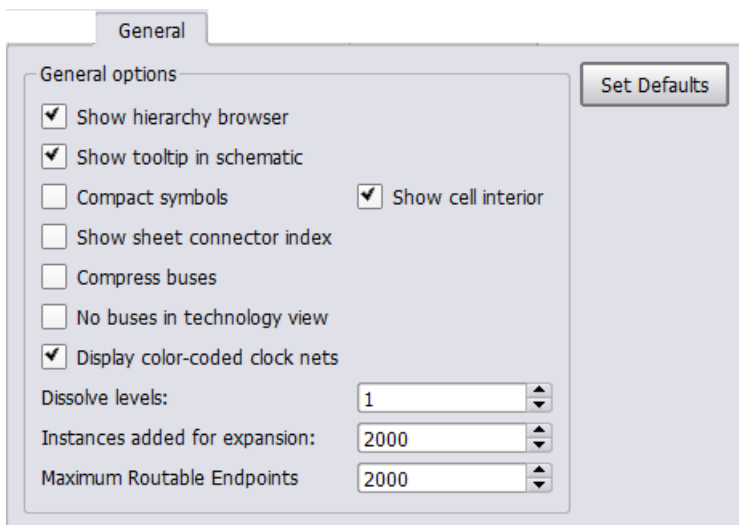
Text Panel



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"> • Show instance name • Show short instance name • No instance name
Set Defaults	Set the dialog box to display the default values.

General Panel



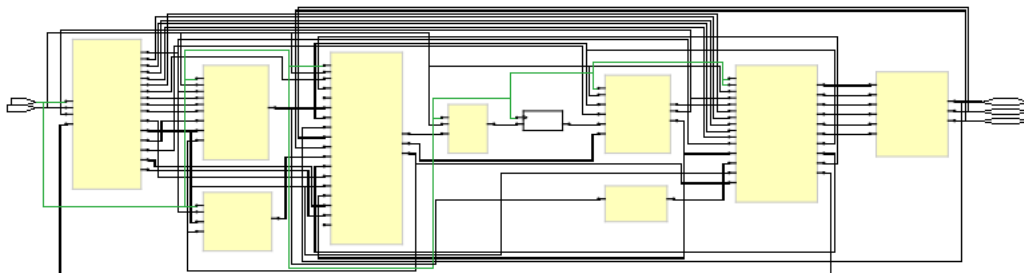
The following options are in the General panel.

Field/Option	Description
Show hierarchy browser	When enabled, a hierarchy browser is present at the left pane of RTL and Technology views.
Show tooltip in schematic	When enabled, displays tooltips that hover objects as you move over them in the RTL and Technology schematic views.
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 Technology views. This is not available if Compact symbols is enabled.
Show sheet connector index	When enabled, sheet connectors show connecting sheet numbers – see Sheet Connectors, on page 99 .

Field/Option	Description
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.
No buses in technology view	When enabled, buses are not displayed, they are only indicated as bits in a Technology View. This applies only to flattened views created by HDL Analyst->Technology->Flattened View (or Flattened to Gates View), not to hierarchical views that you have flattened (using, for example, HDL Analyst->Flatten Current Schematic).
Display color-coded clock nets	Displays clock nets in the HDL Analyst View with the color green. See Color-coded Clock Nets, on page 333 .
Dissolve levels	The number of levels to dissolve, during HDL Analyst->Dissolve Instances. See Dissolve Instances, on page 312 .
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 Results of Maximum Routable Endpoints in the HDL Analyst View, on page 334 .

Color-coded Clock Nets

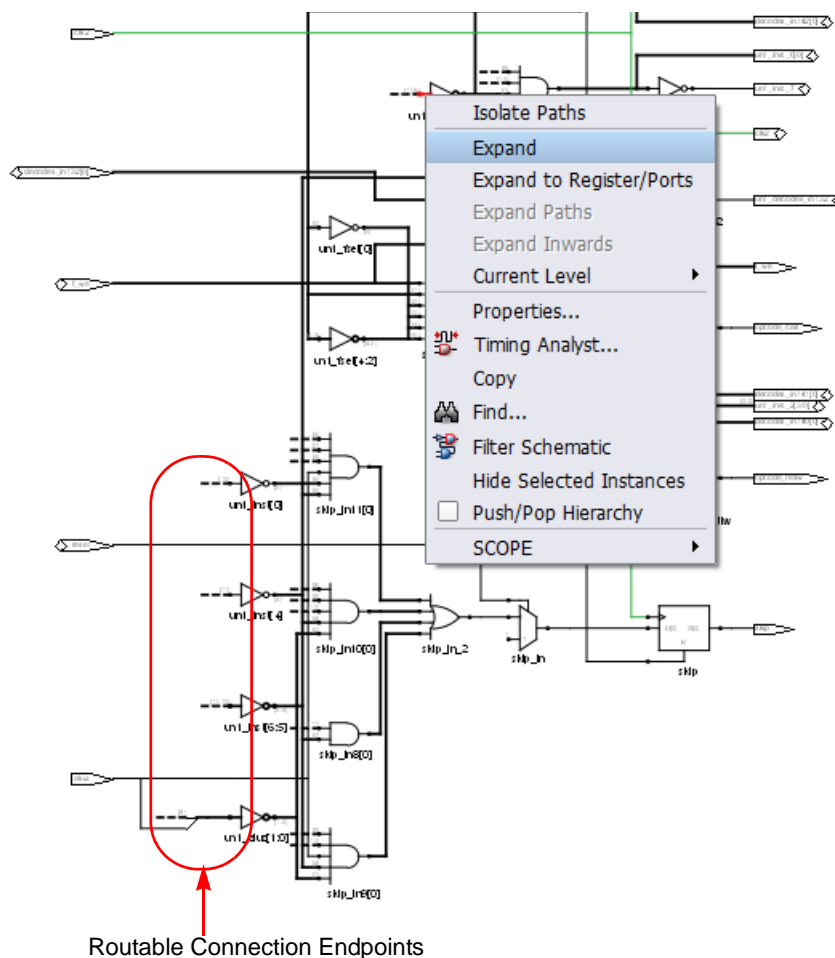
Clock nets are displayed with the color green in the RTL and Technology views.



Results of Maximum Routable Endpoints in the HDL Analyst View

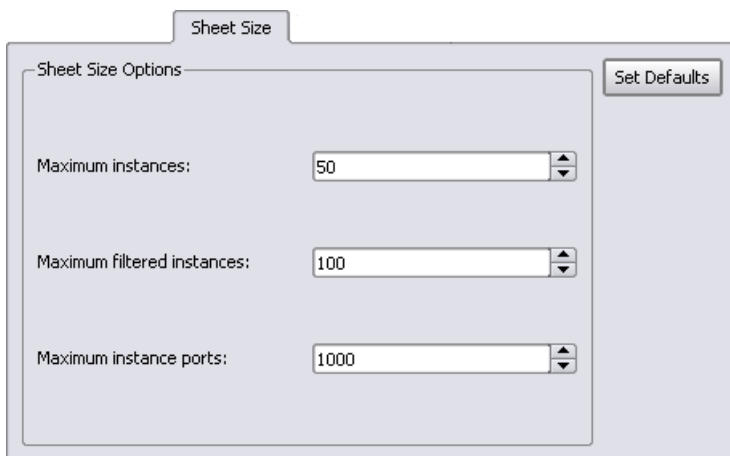
Use the Maximum Routable Endpoints option to specify the maximum number of endpoints for nets in the design to be explicitly routed to their connection endpoints. When you adjust the default value of 2000 sufficiently, improvements in performance can be seen in the HDL Analyst tool.

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



Note: Occasionally, the software does not route nets for some reason. You will see dashes (---) for these endpoints in the HDL Analyst (RTL or Technology) 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 section titled "Sheet Size Options" with three spinners and a "Set Defaults" button.

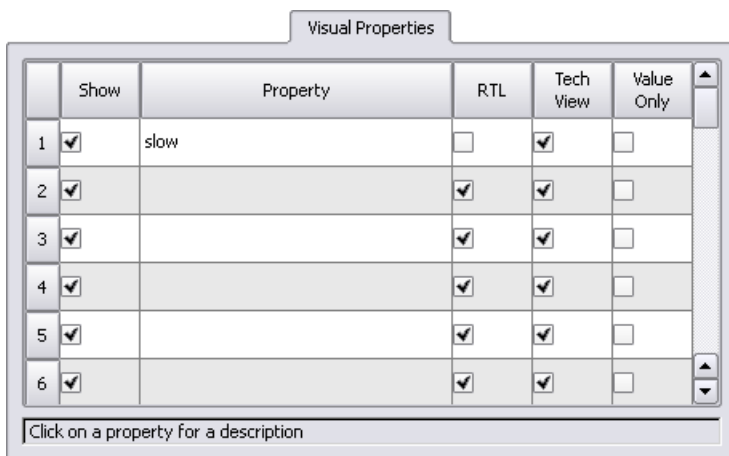
Option	Value
Maximum instances:	50
Maximum filtered instances:	100
Maximum instance ports:	1000

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. See Multiple-sheet Schematics, on page 112 .
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</p> <ul style="list-style-type: none">• the sheet itself is a level, and• each transparent instance is a level (even if inside another transparent instance). <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>(Only children are counted, not grandchildren or below. Instance A is a <i>child</i> of instance B if it is inside no other instance that is inside B.)</p> <p>In fact, 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. See Multiple-sheet Schematics, on page 112.</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 257](#) in the *User Guide*.

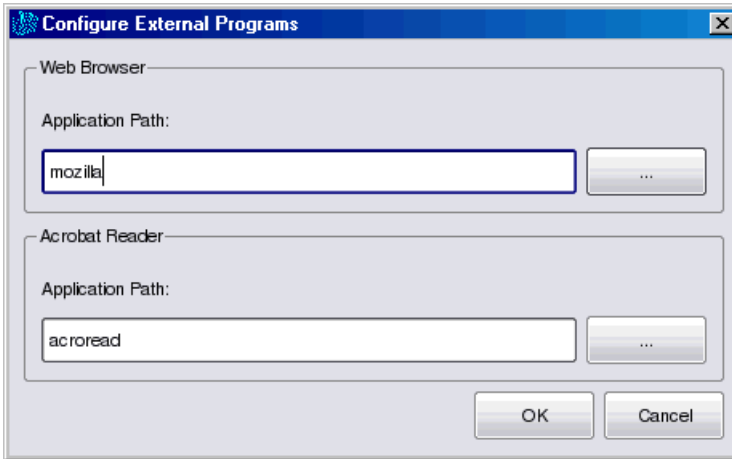


The following options are in the Visual Properties panel.

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 RTL view.
Tech View	Enables or disables the display of visual properties of in the Technology view.
Value Only	Displays only the value of an item and not its property name.

Configure External Programs Command

This command is for Linux platforms only. It lets you specify the web browser and PDF reader for accessing Synopsys support (see [Web Menu, on page 340](#) for details) and online documentation.



Field/Option	Description
Web Browser	Specify your web browser as an absolute path. You can use the Browse button to locate the browser you need. The default is netscape. If your browser requires additional environment settings, you must do so outside the synthesis tool.
Acrobat Reader	Specify your PDF reader as an absolute path. You can use the Browse button to locate the reader you need. The default is acroread.

Web Menu

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

Command	Description
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

There are four help systems accessible from the Help menu:

- Help on the Synopsys FPGA synthesis tool (Help->Help Topics)
- Help on standard Tcl commands (Help->TCL)
- Help on using messages (Help->Error Messages)
- Help on using online help (Help->How to Use Help)

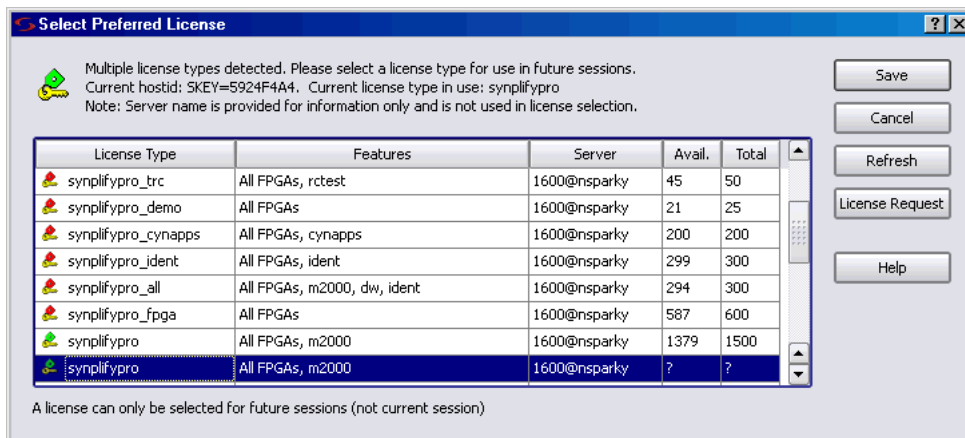
The following table describes the Help menu commands.

Command	Description
Help Topics	Displays hyperlinked online help for the product.
Additional Products	Displays the Synopsys FPGA family of products and a brief description.
How to Use Help	Displays help on how to use Synopsys FPGA online help.
PDF Documents	Displays an Open dialog box with hyperlinked PDF documentation on the product including release notes, user guide, reference manual, and licensing configuration and setup. You need Adobe Acrobat Reader® to view the PDF files.
Error Messages	Displays help on the message viewer.
TCL	Displays help for Tcl commands.
Tutorial	Opens the appropriate tutorial for the synthesis tool you are using.
Mouse Stroke Tutor	Displays the Mouse Stroke Tutor dialog box which provides information on the available mouse strokes – see Using Mouse Strokes, on page 70 for details.
Interactive Attribute Example	Displays the Attribute wizard.
License Agreement	Displays the Synopsys FPGA software license agreement.
Floating License Usage	Specifies the number of floating licenses currently being used and their users.

Command	Description
Preferred License Selection	Displays the floating licenses that are available for your selection. See Preferred License Selection Command, on page 342 .
Tip of the Day	Displays a daily tip on how to use the Synopsys FPGA synthesis tools better. See Tip of the Day Command, on page 343 .
 About this program	Displays the About dialog box, showing the synthesis tool product name, license expiration date, customer identification number, version number, and copyright. 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.

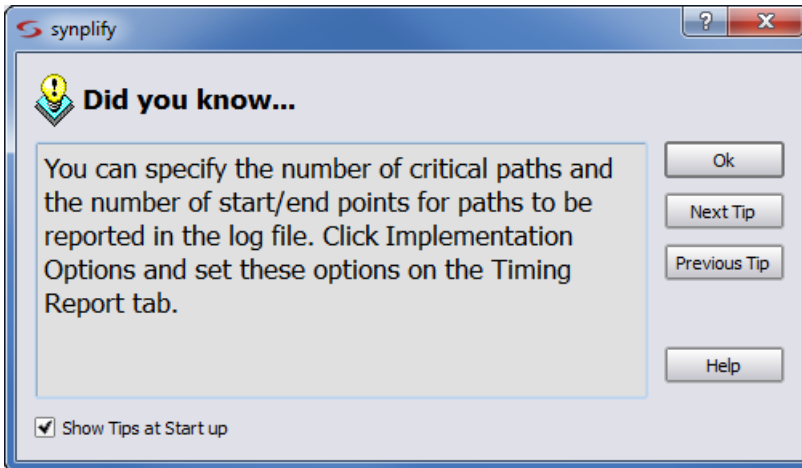
Preferred License Selection Command

Select Help->Preferred License to display the Select Preferred License dialog box, listing the available licenses for you to choose from. Select a license from the License Type column and click Save. Close and restart the Synopsys tool. The new session uses the preferred license you selected.



Tip of the Day Command

Select Help->Tip of the Day to display the Tip of the Day dialog box, with a daily tip on how to best use the Synopsys FPGA synthesis tool. This dialog box also displays automatically when you first start the tool. To prevent it from redisplaying at product startup, deselect Show Tips at Startup.



CHAPTER 5

GUI Popup Menu Commands

In addition to the GUI menu commands described in [Chapter 4, *User Interface Commands*](#), the FPGA synthesis tools also have context-sensitive commands that are accessed from popup or right-click menus in different parts of the interface. Most of these commands have an equivalent menu command. This chapter only describes the unique commands that are not documented in the previous chapter.

See the following sections for details:

- [Popup Menus, on page 346](#)
- [Project View Popup Menus, on page 353](#)
- [RTL and Technology Views Popup Menus, on page 368](#)

Popup Menus

Popup menus, available by clicking the right mouse button, offer quick ways to access commonly used menu commands that are specific to the view where you click. Commands shown greyed out (dimmed) are currently inaccessible. Popup menu commands generally duplicate commands available from the regular menus, but sometimes have commands that are only available from the popup menu. The following table lists the popup menus:

Popup Menu	Description
Project view	See Project View Popup Menus, on page 353 for details.
SCOPE window	Contains commonly used commands from the Edit menu.
Watch Window	See Watch Window Popup Menu, on page 346 for details.
Tcl window	Contains commands from the Edit menu. For details, see Edit Menu Commands for the Text Editor, on page 178 .
Text Editor window	See Text Editor Popup Menu, on page 347 for more information.
RTL and Technology views	See RTL and Technology Views Popup Menus, on page 368 .
FSM viewer	See FSM Viewer Popup Menu, on page 350 .

Watch Window Popup Menu

The Watch window popup menu contains the following commands:

Command	Description
Configure Watch	Displays the Log Watch Configuration dialog box, where you choose the implementations to watch.
Refresh	Refreshes (updates) the window display.
Clear Parameters	Empties the Watch window.

For more information on the Watch window and the Configure Watch dialog box, see [Watch Window, on page 44](#).

Tcl Window Popup Menu

The Tcl window popup menu contains the Copy, Paste, and Find commands from the Edit menu, as well as the Clear command, which empties the Tcl window. For information on the Edit menu commands available in the Tcl window, see [Edit Menu Commands for the Text Editor, on page 178](#).

Text Editor Popup Menu

The popup menu in the Text Editor window contains the following commonly used text-editing commands from the Edit menu: Undo, Redo, Cut, Copy, Paste, and Toggle Bookmark. In addition, HDL Analyst specific commands appear when both an HDL Analyst view and it's corresponding HDL source file is open. For details of these commands, see [Edit Menu Commands for the Text Editor, on page 178](#) and [HDL Analyst Menu, on page 303](#).

The following table lists the commands that are unique to the popup menu:

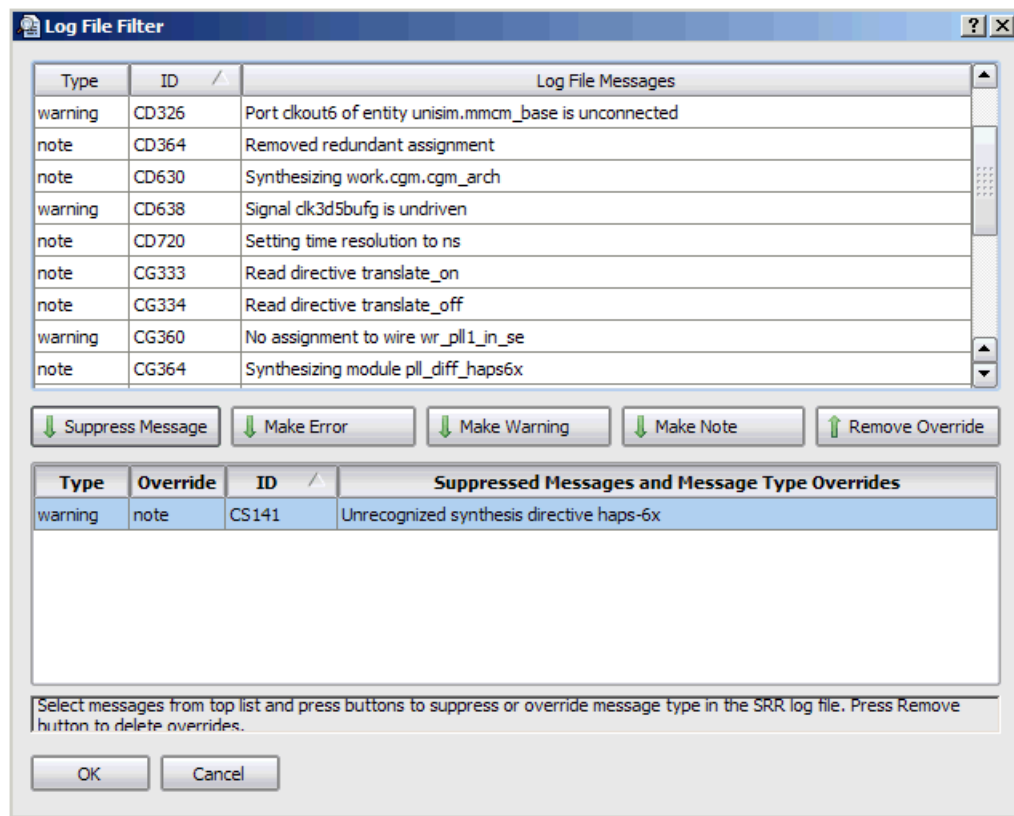
Command	Description
Filter Analyst	Filters your design to show only the currently selected objects in the HDL text file. This is the same as HDL Analyst->Filter Schematic.
Select in Analyst	Crossprobes from the Text Editor and selects the objects in the HDL Analyst view. To use this command, the Enhanced Text Crossprobing (option must be engaged).

Log File Popup Menu

The popup menu in the log file contains commands that control operations in the log file. The popup menu differs when the log file is opened in the HTML mode or in the ASCII text mode.

Log File Filter Dialog Box

The Log File Filter dialog box is available by selecting Log File Message Filter from the log file popup menu when the log file is opened in the HDML mode. The dialog box allows messages in the current session to be promoted or demoted in severity or suppressed from the log files for subsequent sessions. For additional information on using this dialog box, see [Log File Message Controls, on page 204](#) of the *User Guide*.



The following table describes the dialog box functions.

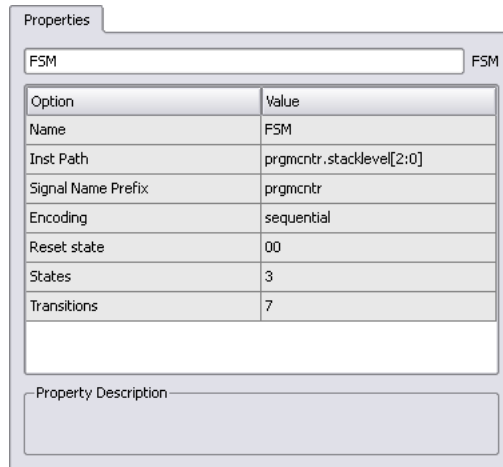
Function	Description
Log File Messages window	Displays the message ID and text and the default message type of messages generated during the current session.
Suppress Message button	Suppresses the selected note, warning, or advisory message. The selected message is removed from the upper Log File Messages window and displayed in the lower window with the Override column indicating suppress status. Note that error messages cannot be suppressed.
Make Error button	Promotes the status of the selected warning (or note) to an error. The selected message is removed from the upper Log File Messages window and displayed in the lower window with the Override column indicating error status.
Make Warning button	Promotes the status of the selected note to a warning. The selected message is removed from the upper Log File Messages window and displayed in the lower window with the Override column indicating warning status.
Make Note button	Demotes the status of the selected warning to a note. The selected message is removed from the upper Log File Messages window and displayed in the lower window with the Override column indicating note status.
Remove Override button	Removes the override status on the selected message in the lower window and returns the message to the upper Log File Messages window.
lower window	Lists the status of all messages that have been promoted, demoted, or suppressed.
OK button	Updates the status of any changed messages in the .pfl file. Note that you must recompile/resynthesize the design before any message status changes become effective.

FSM Viewer Popup Menu

The popup menu in the FSM Viewer contains commands that determine what is shown in the FSM Viewer. The following table lists the popup commands in the FSM Viewer.

Command	Description
Properties	Displays the Object Properties dialog box and view properties of a selected state or transition. Information about a selected transition includes the conditions enabling the transition and the identities of its origin and destination states. Information about a selected state includes its name, RTL encoding, and mapped encoding.
Filter	See View Menu: FSM Viewer Commands, on page 193 .
Unfilter	See View Menu: FSM Viewer Commands, on page 193 .
FSM Properties	Displays the Object Properties dialog box indicating the FSM identity and location, encoding style, reset state, and the number of states and transitions.

FSM Properties



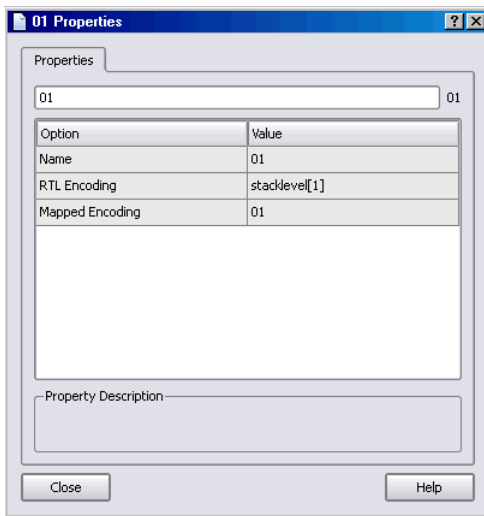
Properties

FSM

Option	Value
Name	FSM
Inst Path	prgmcntr.stacklevel[2:0]
Signal Name Prefix	prgmcntr
Encoding	sequential
Reset state	00
States	3
Transitions	7

Property Description

State properties
(state selected)



01 Properties

Properties

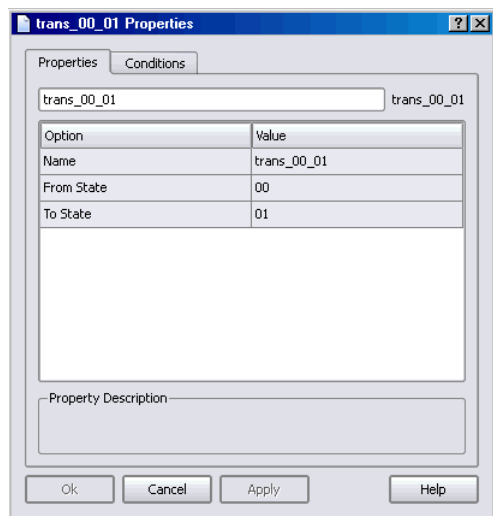
01

Option	Value
Name	01
RTL Encoding	stacklevel[1]
Mapped Encoding	01

Property Description

Close Help

Transition properties
(transition selected)



trans_00_01 Properties




Properties Conditions

trans_00_01

Option	Value
Name	trans_00_01
From State	00
To State	01

Property Description

Ok Cancel Apply Help

Field/Option	Description
  	Icons indicating the object type: FSM, state, or transition.
Name	The name of the selected state or transition, or FSM if nothing is selected.
Inst Path	The full name and position of the state machine in the hierarchy.
Signal Name Prefix	The position of the state machine in the hierarchy.
Encoding	The style of encoding used for the state machine. This can be onehot, sequential, gray, or safe. See syn_encoding , on page 57 , for information on changing the encoding type.
Reset State	The initial state of the FSM: the active state after resetting.
States	The number of states in the state machine.
Transitions	The number of transitions in the state machine.
RTL Encoding	The name (address) of the selected state, as referred to in the RTL (HDL) file.
Mapped Encoding	The encoding of the selected state.
From	The origin state of the selected transition.
To	The destination state of the selected transition.
Conditions (min-terms)	The conditions enabling the selected transition, as defined in the RTL (HDL) file.

Project View Popup Menus

The popup menu commands available in the Project view are context-sensitive, depending on what is currently selected and where in the view you click to open the popup menu. Most commands duplicate commands from the File, Project, Run, and Options menus.

Project Management View Popup Commands

The following table describes the Project Management view commands that are not duplicated on other menus in the tool:

Command	Description
Project Management View, No Selections	
Open Project	Displays the Open Project Dialog. See Open Project Command, on page 176 .
New Project	Creates a new empty project in the Project Window.
Refresh	Refreshes the display.
Project View Options	Displays the Project View Options dialog. See Project View Options Command, on page 319 .
Project Selected	
Open as Text	Opens the selected file in the Text Editor.
Add File	Displays the Add Files to Project dialog. See Add Source File Command, on page 200 .
Synthesize	Compiles and maps the selected design.
Compile Only	Compiles the selected design.
Write Output Netlist Only	Writes the mapped output netlist to structural Verilog (vm) or VHDL (vhm) format. Same as enabling: <ul style="list-style-type: none"> • Write Mapped Verilog Netlist • Write Mapped VHDL Netlist on the Implementation Results tab of the Implementation Options dialog box.
Arrange VHDL Files	Reorders the VHDL source files.

Command	Description
Save Project	Displays the Save Project As dialog box.
Close Project	Closes your project.
Project Folder or File Selected	
Add Folder	Creates a folder with the new name you specified and adds it to the Project view. See Add Folder Command, on page 357 .
Rename Folder	Renames an existing folder with the new name you specified in the Project view. See Rename Folder Command, on page 357 .
Delete Folder	Deletes the specified folder and all its contents as necessary. See Delete Folder Command, on page 358 .
Remove from Folder	Removes the selected file from its corresponding folder.
Place in Folder	Places the selected file into the folder you specify.
Constraint File Selected	
File Options	Displays the File Options dialog box. See File Options Popup Menu Command, on page 359 .
Open	Opens the SCOPE window.
Open as Text	Opens the selected file in the Text Editor.
Copy File	Displays the Copy File dialog box, where you copy the selected file and add it to the current project. You specify a new name for the file. See Copy File Popup Menu Command, on page 361 .
Change File	Opens the Source File dialog box where you choose a new file to replace the selected file. See Change File Command, on page 204 .
Remove File From Project	Removes the file from the project.
HDL File Selected	
File Options	Displays the File Options dialog box. See File Options Popup Menu Command, on page 359 .
Open	Opens the SCOPE window.
Syntax Check	Runs a syntax check on your design code. Reports errors, warnings, or notes in the Tcl Window.

Command	Description
Synthesis Check	Runs a synthesis check on your design code. This includes a syntax check and a check to see if the synthesis tool could map the design to the hardware. No optimizations are performed. Reports errors, warnings, or notes in the Tcl Window.
Copy File	Displays the Copy File dialog box, where you copy the selected file and add it to the current project. You specify a new name for the file. See Copy File Popup Menu Command , on page 361.
Change File	Opens the Source File dialog box where you choose a new file to replace the selected file. See Change File Command , on page 204.
Remove File From Project	Removes the file from the project.
Implementation Selected	
Implementation Options	Displays the Implementation Options dialog box. See Implementation Options Command , on page 214.
Change Implementation Name	Displays the Implementation Name dialog box, where you rename the selected implementation. See Change Implementation Popup Menu Commands , on page 361.
Copy Implementation	Copies the selected implementation and adds it to the current project with the name you specify in the dialog box. See Change Implementation Popup Menu Commands , on page 361.
Remove Implementation	Removes the selected implementation from the project.
RTL View	Creates an RTL View based on the properties of the selected implementation.
Tech View	Creates a Technology View based on the properties of the selected implementation.
Add P&R Implementation	Displays the Add New Place & Route Task dialog box where you set options to run place & route after synthesis. See Add P&R Implementation Popup Menu Command , on page 363
Run	Starts a synthesis run on your design.

Command	Description
---------	-------------

Identify Implementation Selected

Identify Instrumentor	Displays the Identify Instrumentor tool, so that you can instrument signals based on the RTL and SRS netlist (after compile). See Working with the Identify Tools, on page 552 .
Launch Identify Debugger	Launches the Identify Debugger tool to debug the instrumented design. See Working with the Identify Tools, on page 552 .

Place & Route Implementation Selected

Add Place & Route Job	Displays the Add New Place & Route Task dialog box, so you can set options and run placement and routing. See Add P&R Implementation Popup Menu Command, on page 363 .
Remove Place & Route Job	Deletes the place-and-route implementation from the project.
Run Place & Route Job	Runs the place-and-route job for the design.
Run Backannotation Only	Only runs backannotation with placement and timing data immediately following placement and routing.

Project Management Commands

The following table lists the popup commands in the Synplify Pro Project Management view that is not available on the tool command menus. The Project Management view consists of two tabs, and the table lists the popup commands available in both tabs.

Command	Description
---------	-------------

Project Management View -> Project Files Tab Popup Commands

All Synplify project management commands	Refer to the table in Project Management View Popup Commands, on page 353 for descriptions of these commands.
--	---

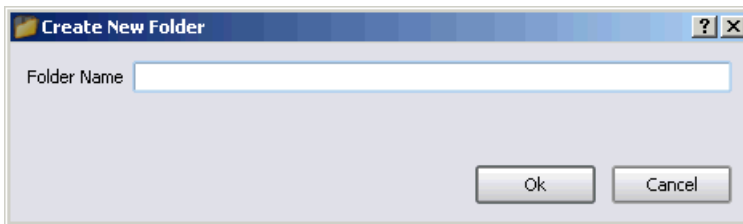
Command	Description
Project Options	With the project selected, displays project properties such as name and location. See Project Options Popup Menu Command , on page 362.
Show Compile Points	Displays the compile points of the selected implementation and lets you edit them. See Show Compile Points Popup Menu Command , on page 362.
P & R Options	With a place-and-route implementation selected, displays the Options for Place & Route on Implementation dialog box, so you can change options and rerun placement and routing. See Options for Place & Route Jobs Popup Menu Command , on page 366 for a description of the features.

Project Management View Popup Folder Commands

The Project view popup menu includes commands for manipulating folders.

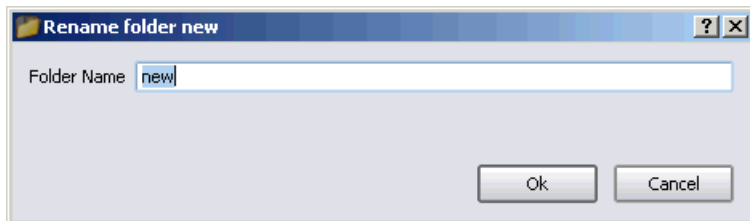
Add Folder Command

Use this option to add a folder to the Project view.



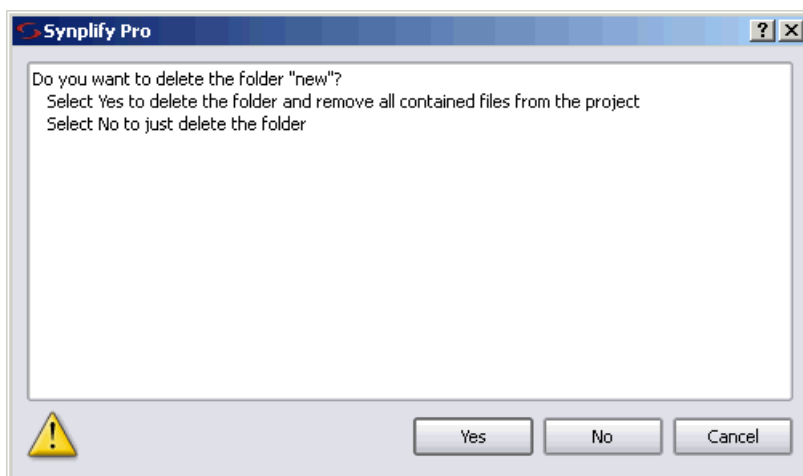
Rename Folder Command

Use this option to rename an existing folder in the Project view.



Delete Folder Command

Use this option to delete a folder from the Project view.



This dialog box includes the following options:

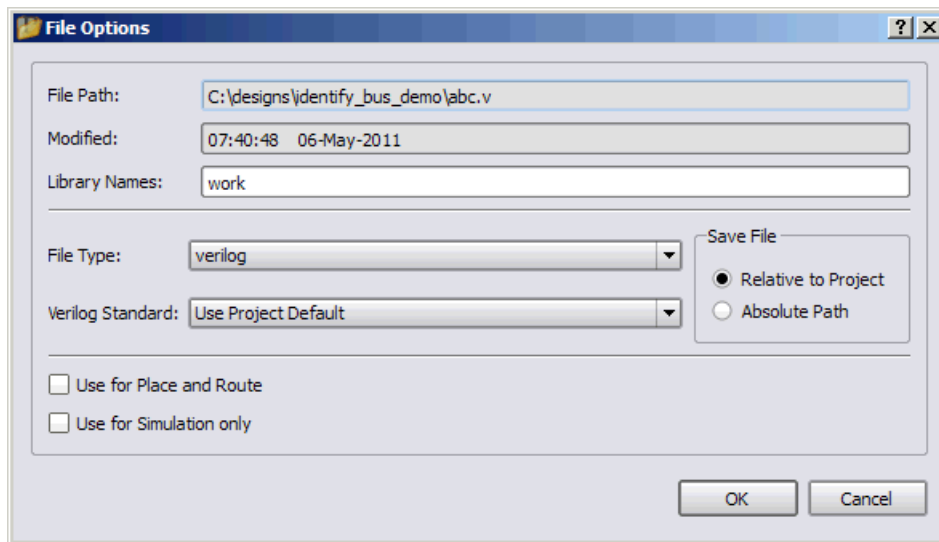
Feature	Description
Yes	Select Yes to delete the folder and all files contained in the folder from the Project view.
No	Select No to delete just the folder from the Project view.
Cancel	Select Cancel, to discontinue the operation.

File Options Popup Menu Command

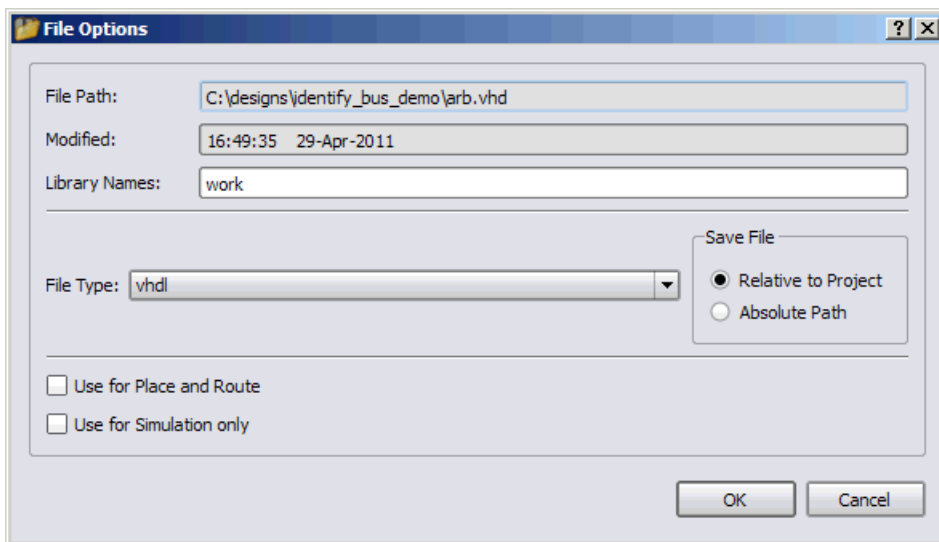
To display the File Options dialog box, right-click on a project file and select File Options from the popup menu. Specify the path as relative or absolute when listing the file in the project (.prj) file and if the file is to be passed to the place-and-route tool or used only for simulation.

Field/Option	Description
File Path	Path to the selected file.
File Type	<p>The folder type for the selected file. You can select the file folder type from a large list of file types.</p> <p>Changing the folder file type does <i>not</i> change the file contents or its extension; it simply places the file in the specified Project view folder. For example, if you change the file type of a VHDL file to Verilog, the file retains its Verilog extension, but is moved from the VHDL folder to the Verilog folder.</p>
Library Names	Name of the library which must be compatible with the HDL simulator. For VHDL files, the dialog box is the same as that accessed by Project->Set VHDL Library – see Set VHDL Library Command, on page 204 .
Last modified	Date the file was last modified.
Save file	The format for the path type: choose either Relative to Project (the default) or with an Absolute Path.
Verilog Standard (Verilog only)	<p>Select the Verilog file type from the menu: Use Project Default, Verilog 95, Verilog 2001, or SystemVerilog.</p> <p>Use Project Default sets the type of the selected file to the default for the project (new projects default to SystemVerilog).</p>
Use for Place and Route	Determines if files are automatically passed to the backend place-and-route tool. The files are copied to the place-and-route implementation directory and then invoked when the place-and-route tool is run.
Use for Simulation Only	Determines if files are only to be used for simulation. For example, files such as test benches containing HDL constructs used only for simulation can be specified using this option.

The following is the Verilog dialog box:

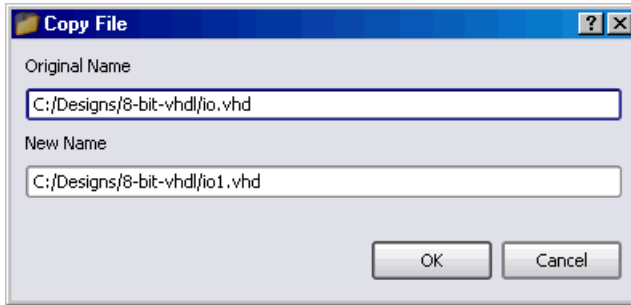


The following is the VHDL dialog box:



Copy File Popup Menu Command

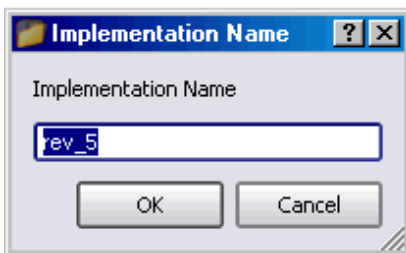
With a file selected, select the Copy File popup menu command to copy the selected file and add it to the current project. This displays the Copy File dialog box where you specify the name of the new file.



Change Implementation Popup Menu Commands

With an implementation selected, right-click and select the Change Implementation Name or Copy Implementation popup menu commands to display a dialog box where you specify the new name.

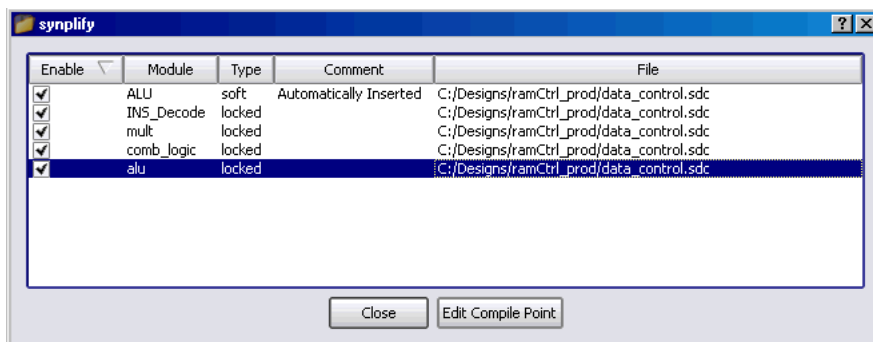
Command	Description
Change Implementation Name	The implementation name you specify is the new name for the implementation.
Copy Implementation	The currently selected implementation is copied and saved to the project with the new implementation name you specify.



Show Compile Points Popup Menu Command

With an implementation selected, select the Show Compile Points popup menu command to display the Compile Points dialog box and view or edit the compile points of the selected implementation.

Compile points are only available for certain technologies. For more information on compile points and the compile-point synthesis flow, see [Compile Point Types, on page 419](#) and [Synthesizing Compile Points, on page 433](#) of the *User Guide*.



The columns Enable, Module, Type, and Comment in the dialog box correspond to the columns Enabled, Module, Type, and Comment in the SCOPE spreadsheet for the compile point. The File column lists the top-level constraint file where the compile point is defined.

To open and edit the SCOPE spreadsheet for a compile point, either double-click the row of the compile point or select it and click the Edit Compile Point button.

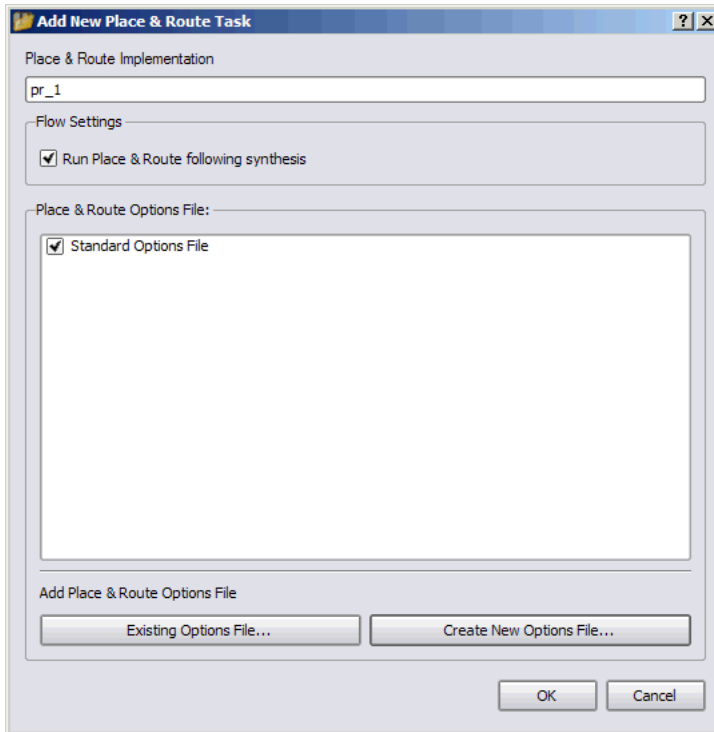
Project Options Popup Menu Command

With a project selected, select the Project Options popup menu command to display the Project Properties dialog box and change the implementation of a project.

In the dialog box, select an implementation in the Implementations list, then click OK or Apply to make it the active implementation of the project.

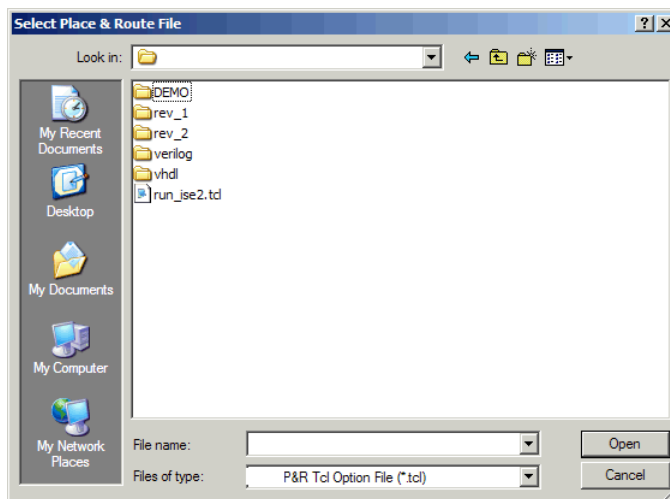
Add P&R Implementation Popup Menu Command

Displays the Add New Place & Route Task dialog box. For information about using this command for place-and-route encapsulation, see [Running P&R Automatically after Synthesis, on page 550](#) in the *User Guide*.



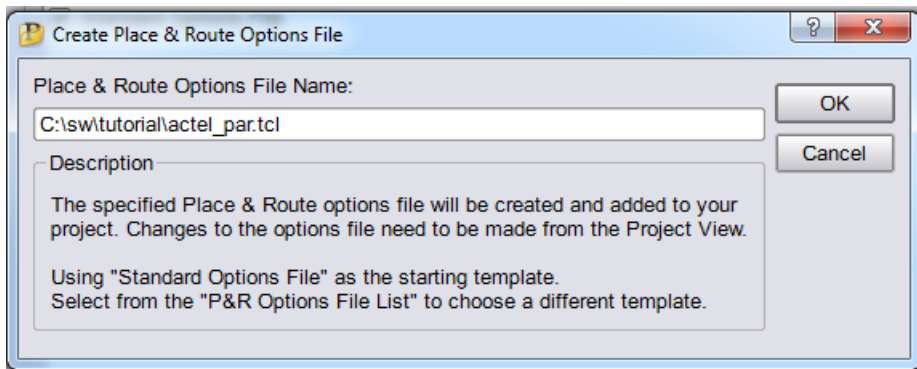
Command	Description
Place & Route Implementation Name	Enter a name for the place & route implementation. Do not use spaces for the implementation name.
Flow Settings	
Run Place & Route following synthesis	Enable/disable the running of the place & route tool from the synthesis tool immediately following synthesis.

Command	Description
Add Place & Route Options File Existing Options File	This option opens the Select Place & Route option file dialog box where you browse for an existing place & route options file. See Running P&R Automatically after Synthesis, on page 550 for information about using this feature.



Add Place & Route Options File Create New Options File	This option opens the Create Place & Route Options File dialog box where you specify a new place & route options file. See Running P&R Automatically after Synthesis, on page 550 for information about creating a new options file.
---	--

Command	Description
---------	-------------

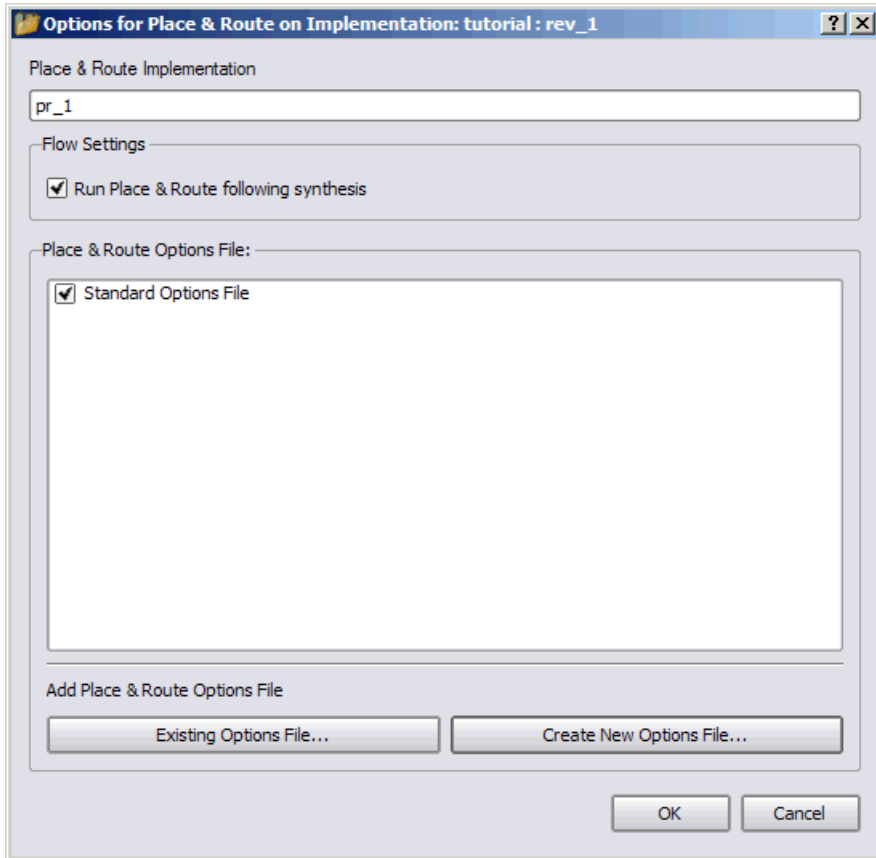


Once the par implementation is created, then you can right-click and perform any of the following options:

- P&R Options—See [Options for Place & Route Jobs Popup Menu Command, on page 366](#).
- Add Place & Route Job—See [Add P&R Implementation Popup Menu Command, on page 363](#).
- Run Place & Route Job—Runs the place-and-route job.

Options for Place & Route Jobs Popup Menu Command

You can select a place-and-route job for a particular implementation, easily change options and then rerun the job. These options are the same found on the Options for Place & Route on Implementation dialog box. For a description of these options, see [Add P&R Implementation Popup Menu Command, on page 363](#).



RTL and Technology Views Popup Menus

Some commands are only available from the popup menus in the RTL and Technology views, but most of the commands are duplicates of commands from the HDL Analyst, Edit, and View menus. The popup menus in the RTL and Technology views are nearly identical. See the following:

- [Hierarchy Browser Popup Menu Commands, on page 368](#)
- [RTL View and Technology View Popup Menu Commands, on page 368](#)

Hierarchy Browser Popup Menu Commands

The following commands become available when you right-click in the Hierarchy Browser of an RTL or Technology view. The Filter, Hide Instances, and Unhide Instances commands are the same as the corresponding commands in the HDL Analyst menu. The following commands are unique to this popup menu.

Command	Description
Collapse All	Collapses all trees in the Hierarchy Browser.
Filter	Highlights and filters objects such as ports, instances, and primitives in the HDL analyst window.
Reload	Refreshes the Hierarchy Browser. Use this if the Hierarchy Browser and schematic view do not match.
Hide/Unhide Instances	Hides or unhides selected instances in the HDL analyst window. For more information on hidden instances, see Hidden Hierarchical Instances, on page 103 .

RTL View and Technology View Popup Menu Commands

The commands on the popup menu are context-sensitive, and vary depending on the object selected, the kind of view, and where you click. In general, if you have a selected object and you right-click in the background, the menu includes global commands as well as selection-specific commands for the objects.

Most of the commands duplicate commands available on the HDL Analyst menu (see [HDL Analyst Menu, on page 303](#)). The following table lists the unique commands.

Common Commands	
Command	See ...
Show Critical Path	HDL Analyst Menu: Timing Commands, on page 310.
Timing Analyst	HDL Analyst Menu: Timing Commands, on page 310.
Find	Find Command (HDL Analyst), on page 183.
Filter Schematic	HDL Analyst Menu: Filtering and Flattening Commands, on page 306.
Push/Pop Hierarchy	HDL Analyst Menu: RTL and Technology View Submenus, on page 303.
Select All Schematic	HDL Analyst Menu: Selection Commands, on page 314.
Select All Sheet	HDL Analyst Menu: Selection Commands, on page 314.
Unselect All	HDL Analyst Menu: Selection Commands, on page 314.
Flatten Schematic	HDL Analyst Menu: Filtering and Flattening Commands, on page 306.
Unflatten Current Schematic	HDL Analyst Menu: Filtering and Flattening Commands, on page 306.
HDL Analyst Options	New HDL Analyst Options Command, on page 329.
SCOPE->Edit Attributes (object <i>name</i>)	Opens a SCOPE window where you can enter attributes for the selected object. It displays the Select Constraint File dialog box (Edit Attributes Popup Menu Command, on page 372), where you select the constraint file to edit. If no constraint file exists, you are prompted to create one.

SCOPE->Edit Compile Point Constraints (module *moduleName*)

For technologies that support compile points, it opens a SCOPE window where you can enter constraints for the selected compile point. It displays the Select Compile Point Definition File dialog box and lets you create or edit a compile-point constraint file for the selected region or instance. See [Edit Attributes Popup Menu Command, on page 372](#).

SCOPE->Edit Module Constraints (module *moduleName*)

Opens a SCOPE window so you can define module constraints for the selected module). If you do not have a constraint file, it prompts you to create one. The file created is a separate, module-level constraint file.

Instance Selected

Command

See ...

Isolate Paths

Isolate Paths, [on page 311](#).

Expand Paths

Hierarchical->Expand Paths, [on page 305](#).

Current Level Expand Paths

Current Level->Expand Paths, [on page 306](#).

Show Context

Show Context, [on page 311](#).

Hide Instance

Hide Instances, [on page 311](#).

Unhide Instance

Unhide Instances, [on page 311](#).

Show All Hier Pins

Show All Hier Pins, [on page 312](#).

Dissolve Instance

Dissolve Instances, [on page 312](#).

Dissolve to Gates

Dissolve to Gates, [on page 312](#).

Port Selected

Command

See ...

Expand to Register/Port

Hierarchical->Expand to Register/Port, [on page 305](#).

Expand Inwards

Hierarchical->Expand Inwards, [on page 305](#).

Current Level->Expand

Current Level->Expand, [on page 305](#).

Current Level->Expand to Register/Port

Current Level->Expand to Register/Port, [on page 306](#).

Current Level->Expand Paths

Current Level->Expand Paths, [on page 306](#).

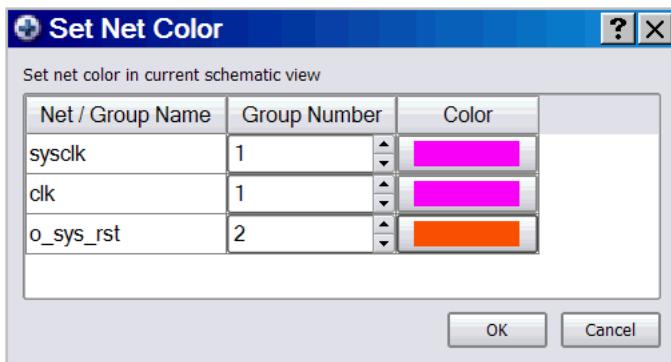
Properties

[Properties Popup Menu Command, on page 372.](#)

Net Selected	
Command	See ...
Goto Net Driver	Hierarchical->Goto Net Driver, on page 305.
Select Net Driver	Hierarchical->Select Net Driver, on page 305.
Select Net Instances	Hierarchical->Select Net Instances, on page 305.
Current Level->Goto Net Driver	Current Level->Goto Net Driver, on page 306.
Current Level->Select Net Driver	Current Level->Select Net Driver, on page 306.
Current Level->Select Net Instances	Current Level->Select Net Instances, on page 306.
Set Net Color	Sets the color of the selected net from a color pallet. For details, see Set Net Color Popup Menu Command, on page 371.

Set Net Color Popup Menu Command

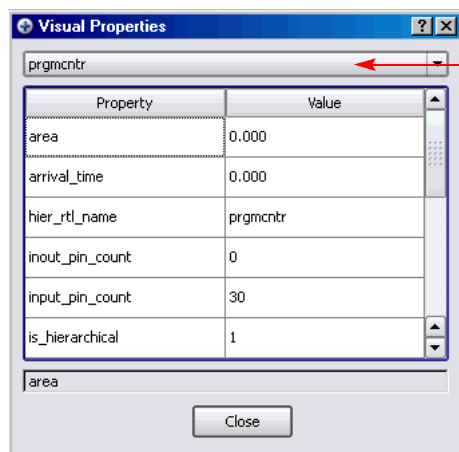
The set net color command sets the color of the selected net in the HDL Analyst for the current session. To use the command, select the desired net or nets in the RTL view and select set net color from the popup menu to display the dialog box.



Double click on the corresponding color in the Color column to display the color pallet and then double click the desired color and click OK. Nets can be grouped and assigned to the same color by selecting the same group number in the Group Number column.

Properties Popup Menu Command

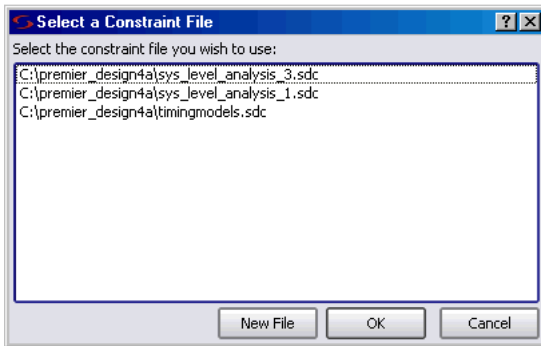
The software displays property information about the selected object when you right-click on a net, instance, pin, or port in a HDL Analyst view. See [Visual Properties Panel, on page 338](#) or [Viewing Object Properties, on page 257](#) in the *User Guide* for more information about viewing object properties.



Lists pins, if the selected object is an instance or net.
Lists bits, if the selected object is a port.

Edit Attributes Popup Menu Command

You use the Select a Constraint File dialog box to choose or create a constraint file. You can open the constraint file and edit it. For technologies that support the compile points, it lets you create or edit a compile-point constraint file for the selected region or instance.



For more information about creating constraint files, see [Using the SCOPE Editor, on page 114](#) of the *User Guide*.

Index

Symbols

- `__ALLOWNESTEDBLOCKCOMMENT-START__` directive [236](#)
- `__SEARCHFILENAMEONLY__` directive [235](#)
- ! character, find command [121](#)
- ? wildcard
 - Timing Analyzer [301](#)
- .srr file
 - See log file
- .srs file
 - See srs file

Numerics

- 64-bit mapping [218](#)

A

- aborting a synthesis run [250](#)
- About this program command [342](#)
- add files
 - _include tcl argument [18](#)
- Add Implementation command [200](#)
- Add P&R Implementation command [363](#)
- Add Place & Route Options File command [364](#)
- Add Place and Route Job command [363](#)
- Add Source File command [199](#)
- add_file Tcl command [17](#)
- add_folder Tcl command [21](#)
- add_to_collection command [160](#)
- Additional Products command [341](#)
- annotated properties for analyst
 - object properties for filtering [117](#)
- append_to_collection command [162](#)
- archive utility
 - `__SEARCHFILENAMEONLY__` directive [235](#)
 - copy tcl command [53](#)
 - unarchive tcl command [53](#)

- Arrange VHDL files command [245](#)
- asynchronous clock report
 - generation option [293](#)
- auto constraints
 - Maximize option (Constraints tab) [219](#)

B

- Back command [193](#)
- batch mode [97](#)
- Build Project command [172](#)
- bus bundling [333](#)
- buses
 - compressed display [333](#)
 - enabling bit range display [331](#)
 - hiding in flattened Technology views [333](#)
- By any transition command [194](#)
- By input transitions command [193](#)
- By output transitions command [193](#)

C

- c_symdiff command, examples [131](#)
- camera mouse pointer [172](#)
- case sensitivity, Tcl find command [108](#), [112](#)
- cell interior display, enabling/disabling [332](#)
- Change File command [199](#)
- Change Implementation Name command [355](#)
- check_fdc_query command [24](#)
- check_fdc_query Tcl command [24](#)
- Clear Parameters command [346](#)
- clock alias [297](#)
- clock as object [297](#)
- Close command [172](#)
- Close Project command [172](#)
- Collapse All command [368](#)
- collection commands
 - c_diff [127](#)
 - c_intersect [128](#)
 - c_list [129](#)

- c_print 130
 - c_syndiff 130
 - c_union 131
 - collections
 - Synopsys standard commands 160
 - commands
 - Add Place & Route Job 363
 - Hierarchy Browser 368
 - menu
 - See individual command entries
 - set_modules (Tcl) 134
 - Tcl
 - See Tcl commands
 - Tcl collection 126
 - Tcl command equivalents 12
 - Tcl expand 123
 - Tcl find 102
 - Comment Code command 178
 - Compile Only command 244
 - compile point constraints
 - editing 370
 - compiler directive
 - _SEARCHFILENAMEONLY_ 235
 - IGNORE_VERILOG_BLACKBOX_GUTS 233
 - compiler directives
 - _ALLOWNESTEDBLOCKCOMMENTS TART_ 236
 - _SYN_COMPATIBLE_INCLUDEPATH_ 239
 - UI option 225
 - Verilog 231
 - Configure External Programs command 339
 - Configure Mapper Parallel Job command 315
 - Configure Verilog Compiler command 315
 - Configure VHDL Compiler command 315
 - Configure Watch command 346
 - connectivity, enabling bit range display 331
 - constraint checker
 - check_fdc_query command 24
 - constraint file
 - syn_create_err_net 95
 - constraint files
 - editing compile point files 370
 - constraint_file Tcl command 26
 - constraints
 - automatic. See auto constraints
 - check constraints 245
 - Constraints panel
 - Implementation Options dialog box 218
 - context-sensitive popup menus
 - See popup menus
 - Continue on Error
 - Configure Compile Point Process 317
 - Continue on Error compile point option 317
 - Copy command 177
 - Copy File command 354
 - Copy Implementation command 355
 - copy_collection command 163
 - copying image
 - Create Image command 172
 - Create Image command 172
 - Create Place & Route Options file dialog box 364
 - critical paths
 - creating new schematics 304
 - custom timing reports 292
 - finding 310
 - Timing Report panel, Implementation Options dialog box 223
 - custom folders
 - project_folder Tcl command 62
 - Customize command 315
 - customizing
 - project files 320
 - Cut command 177
- ## D
- Delete all bookmarks command 178
 - design parameters (Verilog)
 - extracting 231
 - Device panel
 - Conservative Register Optimization 76
 - Implementation Options dialog box 215
 - dialog boxes
 - Implementation Options 214
 - directives
 - _SEARCHFILENAMEONLY_ 235
 - beta features 234
 - ignore syntax check 233
 - IGNORE_VERILOG_BLACKBOX_GUTS 233
 - specifying for the compiler (Verilog) 231
 - display settings
 - Project view 320
 - Dissolve Instances command 312
 - Dissolve to Gates command 312

- dissolving instances [312](#)
- duplicate modules (Verilog)
 - Tcl option [71](#)

E

- Edit Attributes command [369](#)
- Edit Compile Point Constraints command [370](#)
- Edit menu [177](#)
 - Advanced submenu [178](#)
- Edit Module Constraints command [370](#)
- Editor Options command [315](#)
- Enable Slack Margin [296](#)
- encoding
 - enumeration, default (VHDL) [229](#)
 - state machine
 - displaying [352](#)
- encryptIP script [36](#)
 - command-line arguments [36](#)
 - output methods [38](#)
 - syntax [36](#)
- encryptP1735 script [40](#)
 - command-line arguments [41](#)
 - syntax [41](#)
- enumeration encoding, default (VHDL) [229](#)
- environment variables
 - accessing, get_env Tcl command [42](#)
- examples
 - Tcl find command syntax [112](#)
- Exit command [173](#)
- Expand command
 - current level [305](#)
 - hierarchical [305](#)
- Expand Inwards command [305](#)
- Expand Paths command
 - current level [306](#)
 - hierarchical [305](#)
- Expand to Register/Port command
 - current level [306](#)
 - hierarchical [305](#)
- expanding
 - paths between schematic objects [305](#)
- Extract Parameters [231](#)

F

- FDC
 - standard collection commands [160](#)
- File menu

- Recent Projects submenu [173](#)
- File Options command [359](#)
- files
 - .ta *See also* timing report file [294](#)
 - adding to project [17](#), [200](#)
 - constraint [26](#)
 - copying [354](#), [355](#)
 - include [18](#)
 - log. *See* log file
 - opening recent project [173](#)
 - organization into folders [320](#)
 - project [61](#)
 - removing from project [199](#)
 - replacing in project [204](#)
 - srs *See* srs file
 - stand-alone timing report (.ta) [291](#)
 - temporary [313](#)
 - timing report. *See also* timing report file [294](#)
- Filter Schematic command [306](#)
 - popup menu [347](#)
- filtering
 - critical paths [310](#)
 - FSM states and transitions [193](#)
 - paths from pins or ports [311](#)
 - selected objects [306](#)
 - timing reports [293](#)
- Find again command [178](#)
- Find command
 - HDL Analyst [183](#)
 - Text Editor [177](#)
- find command
 - batch mode [50](#)
 - filter properties [117](#)
- finding
 - critical paths [310](#)
- Flatten Current Schematic command
 - filtered schematic [308](#)
 - unfiltered schematic [307](#)
- Flattened Critical Path command [304](#)
- flattened schematic, creating [304](#)
- Flattened to Gates View command [304](#)
- Flattened View command [304](#)
- flattening
 - instances [312](#)
 - schematics [307](#)
- Floating License Usage command [341](#)
- folders
 - adding to project [21](#)

- folders for project files [320](#)
- foreach_in_collection command [164](#)
- Forward command [193](#)
- FPGA Implementation Tools command [340](#)
- from points
 - object search order (Timing Analyzer) [297](#)
 - timing analyzer [297](#)
- FSM Explorer command [245](#)
- FSM Table command [194](#)
- FSM Viewer
 - popup menu [350](#)
 - popup menu commands [350](#)
- FSMs
 - optimizing with FSM Compiler [79](#)
- Full View command [192](#)

G

- get_env Tcl command [42](#)
- get_object_name command [165](#)
- get_option Tcl command [42](#)
- Goto command [178](#)
- Goto Net Driver command
 - current level [306](#)
 - hierarchical [305](#)
- gui
 - synthesis software [10](#)

H

- HDL Analyst
 - Find command [183](#)
 - Visual Properties [193](#)
- HDL Analyst menu [303](#)
 - Current Level submenu [305](#)
 - Hierarchical submenu [305](#)
 - RTL submenu [304](#)
 - Select All Schematic submenu [314](#)
 - Select All Sheet submenu [314](#)
 - Technology submenu [304](#)
- HDL Analyst Options command [316](#)
- HDL Analyst tool
 - displaying timing information [310](#)
- HDL parameter overrides [44](#)
- hdl_define Tcl command [43](#)
- hdl_param Tcl command [44](#)
- Help command [341](#)
- Help menu [341](#)

- Hide Instances command [311](#)
- hiding instances [311](#)
- Hierarchical Critical Path command [304](#)
- Hierarchical View command [304](#)
- hierarchy
 - flattening [307](#)
- Hierarchy Browser
 - commands [368](#)
 - popup menu [368](#)
 - refreshing [368](#)
- hierarchy browser
 - enabling/disabling display [332](#)
- High Reliability panel
 - Implementation Options dialog box [223](#)
- How to Use Help command [341](#)

I

- impl Tcl command [45](#)
- implementation options
 - Options Panel [217](#)
- Implementation Options command [199](#), [214](#)
- Implementation Options dialog box [205](#), [214](#)
 - Constraints panel [218](#)
 - Device panel [215](#)
 - High Reliability panel [223](#)
 - Options panel [217](#)
 - Place and Route panel [242](#)
 - Timing Report panel [222](#)
 - Verilog panel [224](#)
 - VHDL panel [228](#)
- implementation options, device
 - partdata tcl command [51](#)
- Implementation Results panel
 - Options for implementation dialog box [220](#)
- implementations
 - creating [200](#)
 - naming [355](#)
- Import IP
 - commands [243](#)
- Import IP menu [243](#)
- include command
 - verilog library directories [227](#)
- include files [18](#)
- index_collection command [166](#)
- instances
 - dissolving [312](#)
 - expanding paths between [305](#)

- expansion maximum limit [333](#)
- expansion maximum limit (per filtered sheet) [337](#)
- expansion maximum limit (per unfiltered sheet) [337](#)
- finding by name [177](#)
- hiding and unhiding [311](#)
- isolating paths through [311](#)
- making transparent [312](#)
- name display [331](#)
- selecting all in schematic [314](#)
- Instances command
 - schematic selection [314](#)
 - sheet selection [314](#)
- IP
 - license queuing syntax [98](#)
- IP core wizard [246](#)
- IP cores (SYNCore)
 - building ram models [252](#)
- Isolate Paths command [311](#)

J

- Job Status command [246](#), [250](#)
- job tcl command [46](#)

L

- labels, displaying [331](#)
- Launch Identify Instrumentor command [250](#), [252](#)
- levels
 - See hierarchy
- license
 - floating [341](#)
 - saving [342](#)
 - specifying in batch mode [97](#)
- License Agreement command [341](#)
- license queuing [99](#)
- Limit Number of Paths [296](#)
- Linux, 64-bit mapping [218](#)
- Log File
 - HTML [196](#)
 - text [196](#)
- log file
 - displaying [192](#)
 - Tcl commands for filtering [100](#)
- Log File command
 - View menu [196](#)
- Log Watch window

- popup menu [346](#)
- Log Watch Window command [192](#)
- log_filter Tcl command
 - syntax [47](#)
- log_report Tcl command [48](#)
- Lowercase command [178](#)

M

- maximum parallel jobs [318](#)
- memory compiler [252](#)
- memory, saving [313](#)
- menubar [10](#)
- Menus
 - Import IP [243](#)
- menus
 - context-sensitive
 - See popup menus
 - Edit [177](#)
 - HDL Analyst [303](#)
 - Help [341](#)
 - Options [315](#)
 - popup
 - See popup menus
 - Project [199](#)
 - Run [244](#)
 - View [191](#)
- Messages
 - Tcl Window command [191](#)
- Mouse Stroke Tutor command [341](#)
- multiple drivers
 - resolving [84](#)
- Multiple File Compilation Unit
 - Verilog panel [225](#)
- multiple projects
 - displaying project files [321](#)
- MultiProcessing
 - Continue on Error mode [317](#)
- multiprocessing
 - maximum parallel jobs [318](#)

N

- net drivers
 - displaying and selecting [305](#)
- netlist formats
 - Implementation Options dialog box,
 - Implementation Results panel [222](#)
- nets

- expanding hierarchically from pins and ports 305
- finding by name 177
- selecting instances on 305

New command 172

New Implementation command 205

New Project command 172

Next Bookmark command 178

Next Error command 247

Next Sheet command 193

Normal View command 192

O

- object prefixes
 - Tcl find command 106, 111
- object properties
 - annotated properties for analyst 117
- object search order (Timing Analyzer) 297
- object types
 - Tcl find command 106, 111
- objects
 - displaying compactly 332
 - expanding paths between 305
 - filtering 306
 - unselecting
 - all in schematic 314
- Online Documents command 341
- Open command
 - File menu 172
- Open Project command 172
- open_design command 50
- open_file command 50
- opening
 - project 172
- operators
 - Tcl collection 126
- option settings
 - reporting 42
- options
 - setting 66
- Options for implementation dialog box
 - Implementation Results panel 220
- Options menu 315
- Options panel
 - Implementation Options dialog box 217
- output files
 - log. *See* log file
 - srs

- See* srs file
- Overview of the Synopsys FPGA Synthesis Tools 10

P

Pan command 192

parameters

- overriding HDL 44
- SYNCore adder/subtractor 275
- SYNCore byte-enable RAM 268
- SYNCore counter 279
- SYNCore FIFO 254
- SYNCore RAM 264
- SYNCore ROM 271

partdata tcl command 51

Paste command 177

path filtering 296

paths

- expanding hierarchically from pins and ports 305

pins

- displaying names 331
- displaying on transparent instances 312
- expanding hierarchically from 305
- expanding paths between 305
- isolating paths from 311
- maximum on schematic sheet 337

place & route

- run from the synthesis tool 363

Place and Route panel

- Implementation Options dialog box 242

place and route tcl commands

- job 46

pointers, mouse

- zoom 192

popup menus

- FSM Viewer 350
- Hierarchy Browser 368
- Log Watch window 346
- Project view 353
- RTL view 368
- Tcl window 347
- Technology view 368

ports

- displaying names 331
- expanding hierarchically from 305
- expanding paths between 305
- finding by name 177
- isolating paths from 311

- selecting all in schematic [314](#)
- Ports command
 - schematic [314](#)
 - sheet [314](#)
- preferences
 - project file display [320](#)
- Preferred License Selection command [342](#)
- prefixes
 - Timing Analyzer points [297](#)
- Previous bookmark command [178](#)
- Previous Error/Warning command [247](#)
- Previous Sheet command [193](#)
- primitives
 - internal logic, displaying [332](#)
- Print command [172](#)
- Print Setup command [172](#)
- printing
 - view [172](#)
- printing image
 - Create Image command [172](#)
- program_terminate command [52](#)
- program_version command [52](#)
- project files
 - organization into folders [320](#)
- Project menu [199](#)
 - commands [199](#)
- Project Options command [357](#)
- project Tcl command [53](#)
- Project view
 - display settings [320](#)
 - popup menu [353](#)
 - setting up [319](#)
- Project View Options command [315](#)
- project_data Tcl command [60](#)
- project_file Tcl command [61](#)
- project_folder
 - Tcl command [62](#)
- projects
 - adding files [200](#)
 - closing [172](#)
 - creating (Build Project) [172](#)
 - creating (New) [172](#)
 - displaying multiple [321](#)
 - opening [172](#)
- properties
 - find command [117](#)
 - project [60](#)
- Push Tristates

- Verilog panel [225](#)
- Push/Pop Hierarchy command [193](#)

Q

- quitting a synthesis run [250](#)

R

- recent projects, opening [173](#)
- recording command [63](#)
- Redo command [177](#)
- Refresh command [346](#)
- regular expressions
 - Tcl find command [106](#), [111](#)
- Reload command [368](#)
- Remove Files From Project command [199](#)
- Remove Implementation command [355](#)
- remove_from_collection command [167](#)
- Replace command
 - Text Editor [178](#)
- replacing
 - text [189](#)
- report_clocks command [64](#)
- reports
 - timing report (.ta file) [291](#)
- Resolve Multiple Drivers option [84](#)
- resource sharing
 - Resource Sharing option [218](#)
- Resynthesize All command [244](#)
- RTL view
 - displaying [50](#)
 - opening hierarchical view [304](#)
 - popup menu [368](#)
 - popup menu commands [368](#)
 - printing [172](#)
- Run All Implementations command [246](#)
- Run menu [244](#)
- Run Tcl Script command [246](#), [247](#)
- running place & route [363](#)

S

- sar file
 - Archive Project command [206](#)
- Save All command [172](#)
- Save As command [172](#)
- Save command [172](#)
- schematic objects

- displaying compactly 332
- expanding paths between 305
- filtering 306
- unselecting all 314
- schematics
 - displaying labels 331
 - flattening 307
 - navigating sheets 192
 - opening hierarchical RTL 304
 - sheet connectors 332
 - unselecting objects 314
- SCOPE spreadsheet
 - popup menu commands 346
- sdc
 - standard sdc collection commands 160
- sdc2fdc utility 65
- Select All command 178
- Select All States command 194
- Select in Analyst command 347
- Select Net Driver command
 - current level 306
 - hierarchical 305
- Select Net Instances command
 - current level 306
 - hierarchical 305
- Select Place & Route option file dialog box 364
- Selected command 193
- Set Library command 199
- Set Slack Margin command 310
- Set VHDL Library command 199
- set_option
 - Resolve Multiple Drivers 84
- set_option Tcl command 66
- settings
 - reporting option 42
- sheet connectors 332
- Show All Hier Pins command 312
- Show Compile Points command 357
- Show Context command 311
- Show Critical Path command 310
- Show Timing Information command 310
- sizeof_collection command 169
- slack
 - margin
 - setting 310
 - slack margin 296
- srm file
 - hidden logic not saved 313
- srr file
 - See log file
- srs file
 - hidden logic not saved
- start/end points
 - Timing Report panel, Implementation Options dialog box 223
- state machines
 - See also FSM Compiler, FSM viewer, FSMs.
 - displaying in FSM viewer 314
- encoding
 - displaying 352
- filtering states and transitions 193
- Status Bar command 191
- status_report Tcl command 91
- stopping a synthesis run 250
- symbols
 - enabling name display 331
 - finding by name 177
- syn_connect 94
- syn_create_err_net 95
- syn_tristatetomux attribute
 - effect of tristate pushing 241
- SYNCore
 - adder/subtractor parameters 275
 - byte-enable RAM parameters 268
 - counter parameters 279
 - FIFO parameters 254
 - RAM parameters 264
 - ROM parameters 271
- SYNCore wizard 246, 252
- Synopsys FPGA implementation tools
 - product information 340
- Synopsys FPGA products 340
- Synopsys FPGA Synthesis Tools
 - overview 10
- Synopsys Home Page command 340
- Synplify Pro tool
 - user interface 10
- synplify_pro command-line command 97
- Syntax Check command 245
- synthesis
 - stopping 250
- Synthesis Check command 245
- synthesis jobs
 - monitoring 250
- synthesis software
 - gui 10

synthesis_off directive, handling [229](#)
synthesis_on directive, handling [229](#)
Synthesize command [244](#)
SystemVerilog [225](#)

T

Tcl

- c_diff collection command [127](#)
- c_intersect collection command [128](#)
- c_list collection command [129](#)
- c_print collection command [130](#)
- c_symdiff collection command [130](#)
- c_union collection command [131](#)
- collection commands [126](#)
- set_modules collection command [134](#)
- verilog argument [17](#)
- vhdl argument [17](#)

Tcl (Tool Command Language) [8](#)

tcl argument

- _include [18](#)

Tcl collection commands [126](#)

- c_diff [127](#)
- c_intersect [128](#)
- c_list [129](#)
- c_print [130](#)
- c_symdiff [130](#)
- c_union [131](#)
- set_modules [134](#)

Tcl collection operators [126](#)

Tcl commands

- add_file [17](#)
- add_folder [21](#)
- constraint_file [26](#)
- get_env [42](#)
- get_option [42](#)
- hdl_param [44](#)
- impl [45](#)
- log file commands [100](#)
- project [53](#)
- project_data [60](#)
- project_file [61](#)
- project_folder [62](#)
- set_option [66](#)

Tcl conventions [8](#)

Tcl expand command [123](#)

Tcl find command [102](#)

- case sensitivity [108](#), [112](#)
- examples [112](#)
- object prefixes [106](#), [111](#)

- object types [106](#), [111](#)

- regular expression syntax [106](#), [111](#)

- special characters [106](#), [111](#)

- syntax [103](#)

- wildcards [106](#), [111](#)

TCL Help command [341](#)

Tcl Script

- Tcl Window command [191](#)

Tcl scripts

- running [246](#), [247](#)

Tcl shell command

- sdc2fdc [65](#)

Tcl window

- popup menu [347](#)

Tcl Window command [191](#)

Technical Resource Center

- specifying PDF reader (UNIX) [339](#)

- specifying web browser (UNIX) [339](#)

Technology view

- creating [304](#)

- popup menu [368](#)

- popup menu commands [368](#)

- printing [172](#)

technology view

- displaying [50](#)

text

- copying, cutting and pasting [177](#)

- replacing [189](#)

Text Editor

- popup menu commands [347](#)

- printing [172](#)

through points

- specifying for timing report [295](#)

timing analyst

- generating report [291](#)

timing analyzer

- wildcards [300](#)

timing constraints

- checking [245](#)

timing information, displaying (HDL Analyst tool) [310](#)

timing report

- asynchronous clock report [293](#)

- defining through points [295](#)

- file (.ta) [291](#)

- specifying slack margin [296](#)

- using path filtering [296](#)

timing report file

- generating custom [292](#)

- stand-alone [294](#)
- Timing Report panel
 - Implementation Options dialog box [222](#)
 - Number of Critical Paths [223](#)
 - Start/End Points [223](#)
- timing reports
 - file. *See* timing report file
 - filtering [293](#)
 - parameters [291](#)
 - stand-alone [291](#)
 - stand-alone (.ta file) [291](#)
- Tip of the Day command [342](#)
- to points [297](#)
 - Timing Analyzer [297](#)
- Toggle bookmark command [178](#)
- Toolbars command [191](#)
- tooltips
 - displaying [195](#)
- transparent instances
 - displaying pins [312](#)
- tristates
 - pushing tristates, description [240](#)
 - pushing tristates, example [240](#)
 - pushing tristates, pros and cons [241](#)

U

- Uncomment Code [178](#)
- Undo command [177](#)
- Unfilter command [194](#)
- unfiltering [311](#)
 - FSM diagram [194](#)
 - schematic [311](#)
- Unflatten Current Schematic command [308](#)
- Unhide Instances command [311](#)
- unhiding hidden instance [311](#)
- UNIX
 - configure external programs [316](#)
- Unselect All command [314](#)
 - View menu (FSM Viewer) [194](#)
- Uppercase command [178](#)
- user interface
 - Synplify Pro tool [10](#)
- utilities
 - sdc2fdc [65](#)

V

- variables

- accessing, get_env Tcl command [42](#)
- reporting [42](#)
- VCS Simulator command [246](#)
- Vendor Constraints
 - Implementation Results panel,
Implementation Options dialog
box [222](#)
 - writing [222](#)
- Verilog
 - 'ifdef and 'define statements [231](#)
 - allow duplicate modules (Tcl option) [71](#)
 - beta features [234](#)
 - compiler, configuring [315](#)
 - extract design parameters [231](#)
 - library directories [227](#)
 - specifying compiler directives [231](#)
- Verilog 2001
 - Verilog panel [225](#)
- verilog argument
 - Tcl [17](#)
- Verilog include files
 - using _SEARCHFILENAMEONLY_
directive [235](#)
- Verilog panel [225](#)
 - Implementation Options dialog box [224](#)
 - Multiple File Compilation Unit [225](#)
 - options [225](#)
 - Push Tristates [225](#)
 - SystemVerilog [225](#)
- version information [342](#)
- VHDL
 - compiler, configuring [315](#)
 - enumeration encoding, default [229](#)
 - ignoring code with synthesis off/on [229](#)
- vhdl argument
 - Tcl [17](#)
- VHDL libraries
 - setting up [204](#)
- VHDL panel
 - Implementation Options dialog box [228](#)
- View FSM command [314](#)
- View FSM Info File command [314](#)
- View Log File command [192](#)
- View menu [191](#)
 - Filter submenu [193](#)
 - Log File command [196](#)
 - RTL and Technology view commands
[192](#)
- View Result File command [192](#)
- View Sheets command [193](#)

Visual Properties command [193](#)

W

web browser

 specifying for UNIX [339](#)

wildcards

 Tcl find command [105](#), [106](#), [111](#)

 text Find [180](#)

 text replacement [189](#)

 timing analyzer [300](#)

Windows, 64-bit mapping [218](#)

Write Output Netlist Only command [245](#)

Z

zoom mouse pointer [192](#)

Zoom Out command [192](#)

