Synopsys FPGA Synthesis
Command Reference Manual

January 2014

https://solvnet.synopsys.com

Copyright Notice and Proprietary Information

Right to Copy Documentation

The license agreement with Synopsys permits licensee to make copies of the documentation for its internal use only.

Each copy shall include all copyrights, trademarks, service marks, and proprietary rights notices, if any. Licensee must assign sequential numbers to all copies. These copies shall contain the following legend on the cover page:

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

Destination Control Statement

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

Disclaimer

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

Registered Trademarks (®)

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

Trademarks (™)

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

Service Marks (sm)

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

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

ARM and AMBA are registered trademarks of ARM Limited.

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

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

Printed in the U.S.A
January 2014

Contents

Chapter 1: Introduction

Chapter 2: Tcl Commands

Chapter 3: Tcl Find, Expand, and Collection Commands

Chapter 4: User Interface Commands

Chapter 5: GUI Popup Menu Commands

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. Throughout the documentation, features and procedures described apply to all tools unless specifically stated otherwise.

This chapter includes the following introductory information:

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. In the Synplify Pro tool, you can also run Tcl scripts through the Tcl window (see Tcl Script Window, on page 56).

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 464 in the *User Guide*:

- Running Batch Mode on a Project File, on page 458
- Running Batch Mode with a Tcl Script, on page 459
- Generating a Job Script, on page 465
- Creating a Tcl Synthesis Script, on page 466
- Using Tcl Variables to Try Different Clock Frequencies, on page 468
- Running Bottom-up Synthesis with a Script, on page 470

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 various tools:

- Graphic User Interface Commands, on page 10
- Tcl Commands, on page 12

Graphic User Interface Commands

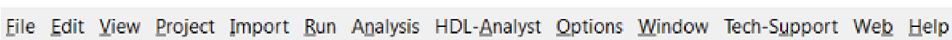
The GUI commands are accessed from the software graphical interface. Most commands open dialog boxes where you can specify parameters for the command.

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 grayed 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 146
- Edit Menu, on page 151
- View Menu, on page 162
- Project Menu, on page 170
- Import Menu, on page 204
- Run Menu, on page 205
- Analysis Menu, on page 252
- HDL Analyst Menu, on page 264
- Options Menu, on page 276
- Tech-Support Menu, on page 296
- Web Menu, on page 301
- Help Menu, on page 305

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 309, Project View Popup Menus, on page 315, and RTL and Technology Views Popup Menus, on page 329 for information on individual popup menus.

Toolbars

Toolbars contain icons associated with commonly used commands. For more information about toolbars, see Toolbars, on page 85.

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 93 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 101 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 Chapter 14, *Scripts*.

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:

- *FPGA Synthesis User Guide*
- *FPGA Synthesis Reference Manual*
- *FPGA Command Reference Manual*
- *FPGA Attributes and Directives 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 86

Alphabetical List of Commands

The commands are listed in alphabetical order. The find, expand, and collection commands appear below, but are described in Tcl Find, Expand, and Collection Commands, on page 89.

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 -vlog_std *standard fileName* [*fileName* [...]]

-filetype	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 18 for a list of the recognized file types.
fileName	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.
-verilog or -vhdl	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 alu.v.new to the project's verilog directory: `% add_file -verilog /designs/megachip/alu.v.new` If you do not specify -verilog, the file is added to the Other directory (new is not a recognized Verilog extension), and the file would not be compiled with the files in the Verilog directory.

[-lib *libName*]	Specifies the library associated with VHDL files. The default library is work. The -lib option sets the VHDL library to *libName*. Note: You can also specify multiple libraries for VHDL files. For example: `add_file -vhdl -lib {mylib,work} "ff.vhd"` Both the logical and physical libraries must be specified in the Project file (if you only specify the logical library associated with the VHDL files, the compiler treats the module as a black box).

[-folder *folderName*]	Creates logical folders with custom files in various hierarchy groupings within your Project view. For example:
	`add_file -verilog -folder memory "ram_1.v"`
	`add_file -verilog -folder memory "C:/examples/verilog/common_rtl/memory/ram_1.v"`

-include	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.
-vlog_std *standard*	Overrides the global Verilog standard for an individual file. The accepted values for *standard* are v95 (Verilog 95), v2001 (Verilog 2001), and sysv (SystemVerilog). The file (*fileName*) 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, commchip.v <sysv>). Note that when you add a SystemVerilog file (a file with an sv extension) to a project, the add_file entry in the project file includes the -vlog_std *standard* string. The default standard for new projects is SystemVerilog. For Verilog 2005 extensions, use sysv (SystemVerilog).

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
.adc	-analysis_constraint	Analysis Design Constraint
.edf, .edn	-edif	EDIF
.fdc	-fpga_constraint/-constraint	Logic Constraints (FDC)
.sdc	-constraint	Logic Constraints (SDC)
.sv[1]	-verilog	Verilog
.tcl	-tcl	Tcl Script
.v	-verilog	Verilog
.vhd, .vhdl	-vhdl	VHDL
any	-include	Include

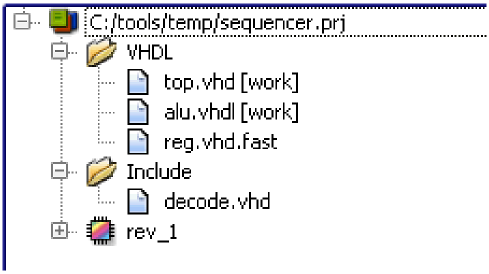
1. Use the .sv format for SystemVerilog keyword support. Both Verilog and SystemVerilog formats are added to the Verilog folder.

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:



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*.

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 -full_check)	Pre-mapping	ram_out [3:0]
With -full_check	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
all_clocks	get_cells
all_inputs	get_clocks
all_outputs	get_nets
all_registers	get_pins
	get_ports

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.

```
FDC query commands results
**************************
################################
# 1019 : set_multicycle_path 2 -from [get_cells -hier {*[4]}]
# line 175 in  :
C:/check_fdc_query/all_clocks/test1_basic/top_translated.fdc
Results of query command: get_cells -hier {*[4]}
   (none)
################################
# 1027 : set_multicycle_path 3 -to [all_clocks]
# line 196 in  :
C:/check_fdc_query/all_clocks/test1_basic/top_translated.fdc
Results of query command: all_clocks
   clka
   clkb
   dcm|CLK0_BUF_clock_CLKIN1
   dcm|clk0_i_clock_CLKIN1
   dcm|CLK0_BUF_1_clock_CLKIN1
```

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

```
# Synopsys Constraint Checker (syntax only), version map610dev,
Build 1085R
# Copyright (C) 1994-2013, Synopsys, Inc.
# Written on Tue Apr 30 15:39:07 2013
##### DESIGN INFO
#######################################################
Top View:                 "top"
Constraint File(s):
"C:\check_fdc_query\all_clocks\test1_basic\top_translated.fdc"

"C:\builds\syn201309_063R\lib\fdc_query.fdc"

# Run constraint checker to find more issues with constraints.
##################################################################
#######

No issues found in constraint syntax.

Clock Summary
**************
Start
Requested    Requested    Clock                    Clock
Clock    Frequency   Period   Type        Group
------------------------------------------------------------------
------------------------------------------------------------------
clka     100.0 MHz   10.000   declared    default_clkgroup
clkb     50.0 MHz    20.000   declared    default_clkgroup
dcm|CLK0_BUF _clock_CLKIN1
         200.0 MHz   5.000    derived (from clka) default_clkgroup
dcm|CLK0_BUF_1 _clock_CLKIN1
         50.0 MHz    20.000   derived (from clka) default_clkgroup
==================================================================
```

See Also

- Constraint Checking, on page 159
- Constraint Checking Report, on page 272

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 Also

- recording, on page 54

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
```

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 662 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	Specifies the symmetric encryption cipher. The keylength must match the algorithm being used, with each character using 8 bits. • 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. See Encryption and Decryption Methodologies, on page 657 in the *Command Reference* for an overview. For information about how symmetric encryption fits into the encryption flow, see Encryption and Decryption, on page 433 in the *User Guide*.
-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	Determines how the IP is treated in the output after synthesis: • 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. See Output Methods for encryptIP, on page 27 for more information.
-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 446 in the *User Guide* for guidelines on how to use these methods effectively.

Output Method	Description
blackbox	With this method, you cannot 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.

Output Method	Description
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.

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 (`vm` or `vhm`), EDIF (`edf` or `edn`), or `vqm`.
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 use by a Synopsys FPGA user.

The encryptP1735 script supports three different use models for encrypting RTL files. See The encryptP1735 Script, on page 658 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; *listofFiles* 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 659 in the *Reference 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
```

export_project

Creates a new module- or instance-based subproject that you can export and insert into the current project. By default, HDL-dependent files are included in the subproject. Use the various options for this command to help you create the subproject easily in batch mode.

Syntax

export_project -module *moduleName* **| -instance** *instanceName*
[**-add_file** *fileName*] [**-filelist** *fileListName*] [**-no_default_hdl**] [**-run_type** *configType*]
[**-project** *projectName*]

Arguments and Options

-module *moduleName*

Specifies the target RTL module for performing module-base subproject export.

-instance *instanceName*

Specifies the target instance for performing instance-based subproject export. You can specify the instance using either FDC notation {i:*insta*} or a hierarchy level such as top.b1.a2. If you do not specify an instance, a module-based subproject and all instances of the module are linked to the exported subproject by default.

-add_file *fileName*
Adds HDL source files to the project. Use either a relative or absolute path, for the source files to be included in your project. These are additional files used in conjunction with the default HDL-dependent files.

-filelist *fileListName*
Specifies a file that contains a list of HDL source files to be included in the subproject. Add one entry per line for each HDL file, specifying either a relative or absolute path to the source files in *fileListName*. When you use this option, the files listed in *fileListName* replace the default files listed in the parent project (prj).

-no_default_hdl
Prevents the automatic adding of HDL-dependent files.

-run_type *configType*
Specifies how to run the subproject for each implementation of the parent project. Choose one of the following configuration modes:

top_down – Compiles the subproject, linking to the top-level project before mapping to a netlist.

bottom_up – Maps the subproject to a netlist, before linking to the top-level project.

-project *projectName*
Specifies the project file name. Use either a relative or absolute path for the export subproject.

Examples

```
export_project -instance b1
export_project -module bblock
export_project -instance c1.b2.a1 -add defines.h
export_project -instance a1 -filelist test_source
   -run_type bottom_up
export_project -module bblock -project ./bblock_inst_b1/bblock.prj
```

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 57 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* **...**]**"**
-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 197 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: • *implName* 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 *model*.
-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.
-type	Specifies the type of implementation to add. For example, the: • -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 *jobType*	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	Description
-option *optionName* [*optionValue*]	Get/set options for *jobName*.
-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 191 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.

Fieldname	Value
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 191 in the *User Guide*.

Syntax

You specify this command after the log_filter commands.

log_report -print *fileName*

Example

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

open_design

The open_design command specifies a netlist file (srs 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 or Technology view. Use open_design to read in the srs or srm file before issuing the find command. See the example below.

Syntax

open_design [**-shared 0**|**1**] *filename*

Where:

- -shared — indicates if the database is to be shared (read-only). Setting this switch to 0 allows the database being searched to be updated which requires additional memory resources.
- *filename* is the RTL (srs) or Technology (srm) file that can be used to search the database.

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 Also

- find, on page 90.

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 *filename*	Loads part file.
-family	Lists available technology families.
-part *family*	Lists all parts in specified family.
-vendor *family*	Returns vendor name for the specified family.

Option	Description
-attribute *attribute family*	Returns the value of the job attribute for the specified family.
-grade [*family*:]*part*	Lists the speed grades available for the specified part.
-package [*family*:]*part*	Lists the packages available for the specified part.
-oem [*family*:]*part*	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 saving any data.

Syntax

program_terminate

Arguments and Options

None

Description

The program_terminate command terminates a tool session without prompting or saving data.

Examples

```
program_terminate
```

program_version

Returns software product version.

Syntax

program_version

Arguments and Options

None

Description

The program_version command returns the software product version number.

Examples

```
% program_version
Synplify Pro I-2013.09
```

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 [**-fg**] [**-all**] [*mode*]

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*]

The following table describes the command options.

Option	Description
-run [**-fg**] [**-all**] [*mode*]	Synthesizes the project, according to the specified options: • **-fg** – Synthesizes in the foreground. • **-all** – Synthesizes all implementations. The *mode* is one of the following keywords: • **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. • **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 -clean option (project -run synthesis -clean), resynthesizes the entire project, including the top level and *all compile points*, whether or not their constraints, implementation options or source code changed since the last synthesis. If not followed by -clean, 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. • **write_netlist** – Writes the mapped output netlist to structural Verilog (vm) or VHDL (vhm) format. You can also use this command in an incremental timing analysis flow. For details, see Run Menu, on page 205 and Generating Custom Timing Reports with STA, on page 277.

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

Option	Description

-archive
-project *filename*
[**-root_dir** *pathname*]
-archive_file *filename***.sar**
-archive_type {**full** | **local** | **customize**}
-add_srs [*fileList*] **-no_input**
	• **project** *filename* – copies a project other than the active project. If you do not use this option, by default the active project is copied.
	• **root_dir** *pathname* – specifies the top-level directory containing the project files.
	• **archive_file** *filename* – is the name of the archived project file.
	• **archive_type** - specifies the type of archive:
	  • full – performs a complete archive; all input and result files are contained in the archive file.
	  • 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 *fileList* 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 examples using the project -archive command, see Project Archive Examples, on page 49.

Option	Description

-copy
-project *filename*
-implement *implementationName*
-dest_dir *pathname*
-copy_type {**full | local | customize**}
-add_srs [*fileList*]
-no_input
	• **project** *filename* – copies a project other than the active project. If you do not use this option, by default the active project is copied.
	• **implement** *implementation_name* – archives all files in the specified implementation.
	• **dest_dir** *directory_pathname* – specifies the directory in which to copy the project files.
	• **copy_type** – specifies the type of file/project copy:
	  • 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 *fileList* 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).

-unarchive **-archive_file** *pathname/filename* **-dest_dir** *pathname*	• **archive_file** *pathname/filename* – is the name of the archived project file. • **dest_dir** *pathname* – specifies the directory in which to write the project files. For examples using the project -unarchive command, see Project Unarchive Example, on page 50.

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.

Insert sub-projects to a top-level design.

```
% project -insert "./block2/block2.prj"
% project -insert "./block3/block3.prj"
% project -insert "./block1/block1.prj"
% project -insert "./control/control.prj"
```

Archive Utility

The archive utility provides a way to archive, extract, or copy your design projects. An archive file is in Synplicity 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 98 in the *User Guide*
Archiving a project for Synplicity technical support	*Tech-Support Menu, on page 296*

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 *projectName* is specified, changes the active project to *projectName*.
-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* **|**
-tooltag *applicationTagName* **|** **- toolargs** [*arguments*] *fileName* }

The following table describes the command options.

Option	Description
-lib	Shows the project file library associated with *fileName*. If *libName* is specified, changes the project file library for the specified file to *libName*.
-name	Shows the project file path for the specified file. If *newPath* is specified, changes t1he location of the specified project file to the directory path specified by *newPath*.

Option	Description
-time	Shows the file time stamp. If a *format* 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 *fileName1* after *fileName2* in HDL file list. If *fileName2* is not specified, moves *fileName1* 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
folderName	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

In the command line:

- **on|off**– turns Tcl command recording on (1) or off (0). 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.

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 *designName*_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.
TclFile	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 466.

sdc2fdc

Translates legacy FPGA timing constraints to Synopsys FPGA timing

Syntax

sdc2fdc

Run it from the Tcl window in the synthesis tool.

See also

- Converting SDC to FDC, on page 150 in the *User Guide*
- sdc2fdc Conversion, on page 154 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 154 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/to
p.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 58.

Option Name	Description
-technology *parameter*	Sets the target technology for the implementation. *parameter* is the string for the vendor architecture. Check the Device panel in the GUI or see Device Panel, on page 184, for a list of supported families.
-part *part_name*	Specifies a part for the implementation. Check the Device panel of the Implementation Options dialog box (see Device Panel, on page 184) for available choices.
-speed_grade -*value*	Sets the speed grade for the implementation. Check the Device panel of the Implementation Options dialog box (see Device Panel, on page 184) for available choices.
-package *value*	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 184) for available choices.
-grade -*value*	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 Technology-specific Tcl Commands, on page 86.

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 58.

analysis_constraint	hdl_define	report_path
areadelay	help	reporting_*type*
area_delay_percent	identify_debug_mode	resolve_multiple_driver
auto_constrain_io	ignore_undefined_libs	resource_sharing
autosm	include_path	result_file
block	job (PR)	retiming
compiler_compatible	libext	run_prop_extract
compiler_constraint	library_path	symbolic_fsm_compiler
constraint	maxfan	synthesis_onoff_pragma
default_enum_encoding	maxfan_hard	top_module
disable_io_insertion	max_parallel_jobs	update_models_cp
dup	multi_file_compilation_unit	use_fsm_explorer
enable64bit	num_critical_paths	vlog_std
fanout_limit	num_startend_points	write_apr_constraint
frequency	opcond	write_verilog
frequency auto	preserve_registers	write_vhdl
globalthreshold	project_relative_includes	

Option	Description	GUI Equivalent
-analysis_constraint *path/filename*.adc	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 *percentValue* **-area_delay_percent** *percentValue*	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

Option	Description	GUI Equivalent
-auto_constrain_io 1|0	Determines whether default constraints are used for I/O ports that do not have user-defined constraints. When disabled, only define_input_delay or define_output_delay constraints are considered during synthesis or forward-annotated after synthesis. When enabled, the software considers any explicit define_input_delay or define_output_delay constraints, as before.	Use clock period for unconstrained IO check box, Constraints Panel
-automatic_compile_point 1|0	Enables/disables the automatic compile point flow, which can analyze a design and identify modules that can automatically be defined as compile points and mapped in parallel using Multiprocessing.	Automatic compile point check box, Options Panel
-autosm 1|0 **-symbolic_fsm_compiler 1|0**	Enables/disables the FSM compiler.	FSM Compiler check box, Options Panel
-beta_vfeatures 1 | 0	Enables/disables the use of Verilog compiler beta features.	Beta Features for Verilog, Verilog Panel
-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.	*Complement* of the Push Tristates Across Process/ Block Boundaries check box, VHDL Panel and Verilog Panel

Option	Description	GUI Equivalent
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 *filename* – 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
continue_on_error 1|0	Supports some Microsemi technologies. The continue_on_error option serves the following function. **Mapper** – when enabled during compile-point synthesis, allows the mapping operation to continue on error and synthesize the remaining compile points. The default for this option (0) is to stop on any compilation or synthesis error.	Continue on Error, Project View checkbox or Options Panel, or Configure Compile Point Process Command
-default_enum_encoding default | onehot | gray | sequential	(VHDL only) Sets the default for enumerated types.	Default Enum Encoding, VHDL panel (see VHDL Panel and Verilog Panel)
-disable_io_insertion 1|0 **-block 1|0**	Enables/disables I/O insertion in some technologies.	Disable I/O Insertion, Device Panel

Option	Description	GUI Equivalent
-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, Verilog Panel
-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 *value* **-maxfan** *value*	Sets the fanout limit guideline for the current project.	Fanout Guide, Device Panel
-frequency *value*	Sets the global frequency.	Frequency, Constraints Panel
-frequency auto	Enables/disables auto constraints.	Auto Constrain, Constraints Panel
-globalthreshold *value*	This option applies only to the following Microsemi technologies: • FUSION • IGLOO/IGLOOE/IGLOO+ • ProASIC3/3E/3L 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.	Device Panel

Option	Description	GUI Equivalent
-hdl_define	For Verilog designs; used for extracting design parameters and entering compiler directives.	Compiler Directives and Design Parameters, Verilog Panel
-**hdl_param**	Shows or sets HDL parameter overrides. See hdl_param, on page 34 for command syntax.	Use this command in the Tcl window of the UI.
-help	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 72.	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.	Select the Identify implementation, then launch: • Launch Identify Instrumentor or • Launch Identify Debugger
-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

Option	Description	GUI Equivalent
-include_path *path*	(Verilog only) Defines the search path used by the ‘include commands in Verilog design files. Argument *path* 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: • 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 project_relative_includes option to update older project files.	Include Path Order, Verilog panel (see Verilog Panel, on page 195)
-job *PR_job_name* **-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.
-libext *.libextName1 .libextName2 ...*	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 43 in the *User Guide*.	Library Extensions (space separated) for each unique file extension, Verilog Panel.

Option	Description	GUI Equivalent
-library_path *directory_pathname*	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 *directory_pathname* 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.	Library Directories on Verilog Panel.
-log_file *logFileName*	Allows you to change the name for a default log file (both the `srr` and `htm` files). For example: `set_option -log_file test` generates the the following files in the Implementation Directory after synthesis is run: • `test.htm` • `synlog\test_premap.srr` • `synlog\test_fpga_mapper.srr` • `synlog\test_fpga_mapper.srr_Min`	Enter command from the Tcl window
loop_limit *loopLimitValue*	Allows you to override the default compiler loop limit value of 2000 in the RTL. You can apply loop limits using the Verilog loop_limit or the VHDL syn_looplimit directive. For details about these directives, see loop_limit, on page 27 and *syn_looplimit*, on page 88 in the *Attribute Reference*.	Loop Limit, Verilog Panel and VHDL Panel.
-maxfan *value* **-fanout_guide** *value*	Sets the fanout limit for the current project. The limit value is a guideline for the tool rather than a hard limit.	Fanout Guide, Device Panel

Option	Description	GUI Equivalent
-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 *n*	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 465 in the *User Guide*.	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.	Verilog Panel
-no_sequential_opt 1|0	Enables or disables the sequential optimizations for the design. (Note that unused registers will still be removed from the design.) The default value is true (sequential optimizations not performed). When true, delay and area size might increase. Value can be 1 or true, 0 or false. With this option enabled, the FSM Compiler is effectively disabled.	Device Panel
-num_critical_paths *value*	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 *value*	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 *value*	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 702 for more information.	Device Panel
-preserve_registers 1|0	This option is available only for Microsemi technologies. 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 *User Guide* for information about updating older project files.	Include Path Order, Verilog Panel

Option	Description	GUI Equivalent
-report_path *integer*	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
-**reporting_***type*	Sets parameters for the stand-alone Timing Analyst report. See Timing Report Parameters for set_option, on page 71 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 73 for details.	Resolve Multiple Drivers, Device Panel
-resource_sharing 1|0	Enables or disables resource sharing. This is a compiler-specific optimization, and does not affect resource sharing in the mapper.	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
-RWCheckOnRam 1 | 0	If read or write conflicts exist for the RAM, enable this option to insert bypass logic around the RAM to prevent simulation mismatch. Disabling this option does not generate bypass logic. For more information about using this option in conjunction with the syn_ramstyle attribute, see syn_ramstyle, on page 144.	Read Write Check on RAM, Device Panel
-supporttypedflt 1|0	When enabled (1), the compiler passes init values through a syn_init property to the mapper. For more information, see VHDL Implicit Data-type Defaults, on page 485.	Implicit Initial Value Support, VHDL Panel
-symbolic_fsm_compiler 1|0 **-autosm 1|0**	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. 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. However, if you turn off sequential optimizations for the design, FSM Compiler and/or the syn_state_machine directive and syn_encoding attribute are effectively disabled. See -no_sequential_opt 1|0 for more information on turning off sequential optimizations.	FSM Compiler check box, Device Panel

Option	Description	GUI Equivalent
-synthesis_onoff_pragma 1|0	Determines whether code between synthesis on/off directives is ignored. When enabled, the software ignores any VHDL code between synthesis_on and synthesis_off directives. It treats these third-party directives like translate_on/ off directives (see translate_off/translate_on, on page 180 for details).	Synthesis on/off Implemented as Translate on/Off, VHDL Panel
-top_module *name*	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 Verilog Panel
-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, Verilog Panel
-write_apr_constraint 1|0	Writes vendor-specific constraint files.	Write Vendor Constraint File, Implementation Results Panel
-write_verilog 1|0 **-write_vhdl 1|0**	Writes Verilog or VHDL mapped netlists.	Write Mapped Verilog/VHDL Netlist, Implementation Results Panel

Timing Report Parameters for set_option

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

async_clock	margin
filename	netlist
filter	output_srm
gen_output_srm	

Reporting Option	Description
-reporting_async_clock	Generates a report for paths that cross between clock groups using the stand-alone Timing Analyst.
-reporting_filename *filename*.**ta**	Specifies the standard timing report file (`ta`) generated from the stand-alone Timing Analyst.
-reporting_filter *filter options*	Generates the standard timing report based on the filter options you specify for paths, such as: • From points • Through points • To points For more information, see: • Timing Report Generation Parameters, on page 253. • Combining Path Filters for the Timing Analyzer, on page 258 • Timing Analyzer Through Points, on page 256. • Specifying From, To, and Through Points, on page 199.
-**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 Option	Description
-reporting_margin *value*	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 *filename*.**srm**	Specifies the associated gate-level netlist file (`srm`) generated from 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 253.

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 fi 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.

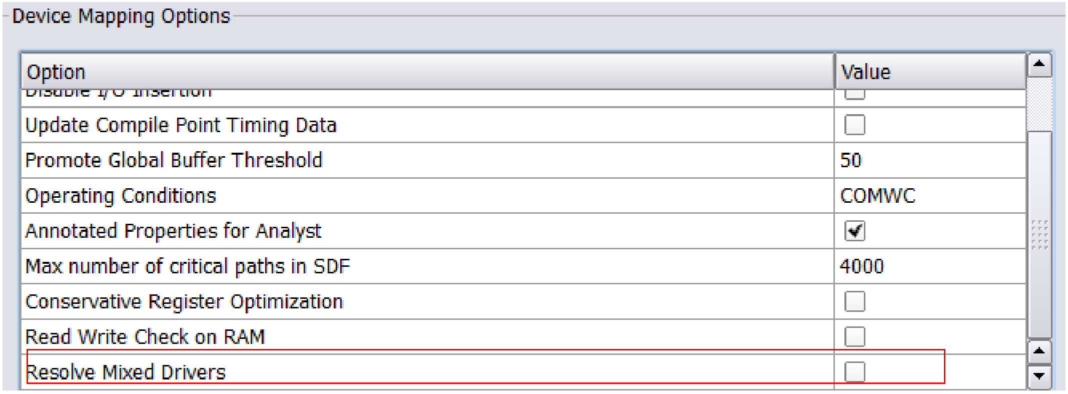
Resolve Mixed Drivers Option

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:dff1.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:

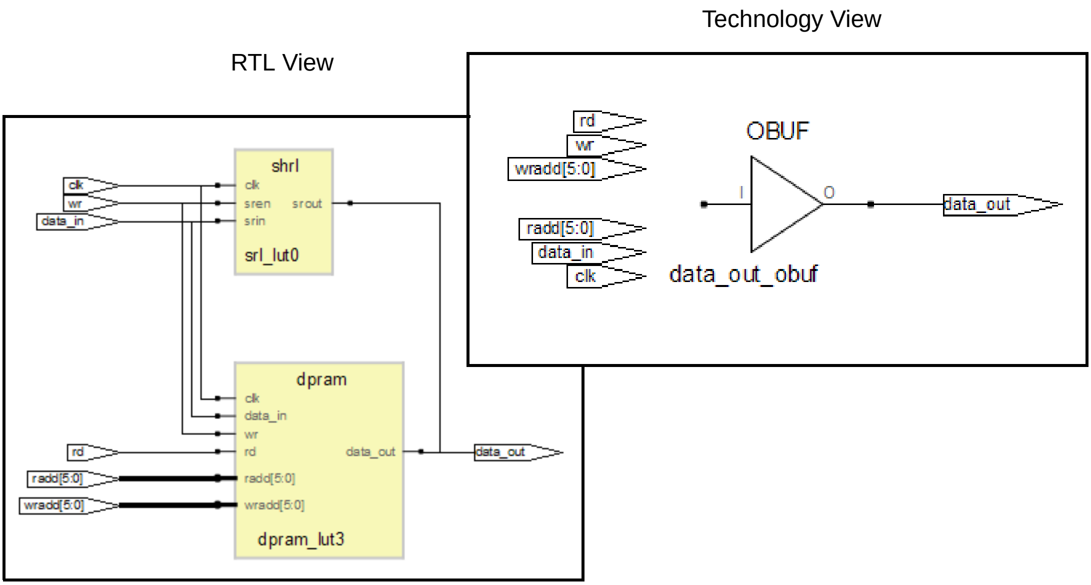
```
@W:BN312 | Resolving mixed driver on net GND, connecting output
pin:data_out inst:dpram_lut3 of work.dpram(verilog) to GND
@N:BN116) | Removing sequential instance dpram_lut3.dout of
view:PrimLib.dffe(prim) because there are no references to its
outputs
@N:BN116 | Removing sequential instance dpram_lut3.mem of
view:PrimLib.ram1(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
shrl 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 shrl (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.

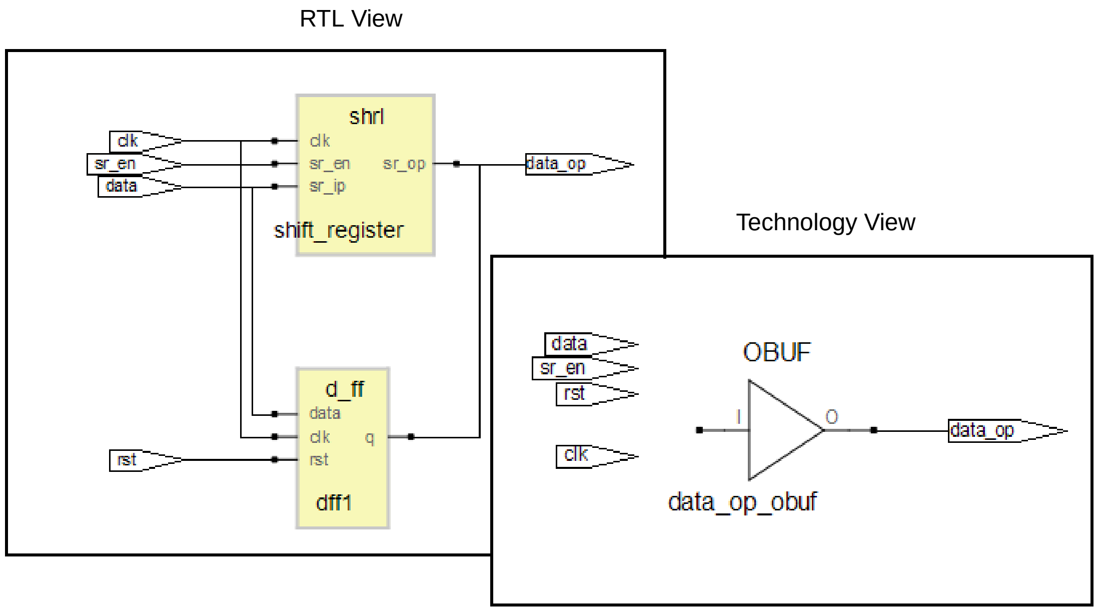

Technology View
RTL View
shrl
clk
sren
srout
srin
srl_lut0
dpram
clk
data_in
wr
rd
data_out
radd[5:0]
wradd[5:0]
dpram_lut3
clk
wr
data_in
rd
radd[5:0]
wradd[5:0]
data_out
OBUF
I
O
data_out
data_out_obuf

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 shrl
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 shrl;

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 : shrl
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 shrl 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 shrl;

architecture rtl of shrl 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.



status_report

This command lets you write out the results of the reports displayed in the Project Status view for the synthesis tools.

Syntax

status_report -name *reportName* [**-parameter** *reportSectionName*]
[**-csv**] [**-output_file** *fileName*] [**-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 timing_report -help
```

Option	Description
-name *reportName*	The name of the report type to access. *reportName* can be any of the following keywords: • area_report • timing_report • opt_report • cp_report • hier_area_report
-parameter *reportSectionName*	Specifies the section of the report for the specific values to access. For details, see Parameters, on page 81.
-csv	Generates the report as a comma separated list.
-output_file *fileName*	Specifies the name of the file to write out the report. If you do not specify an output file, the report is displayed in the Tcl window.
-help	Allows you to get help on a parameter list. Use -help * for a list of parameters.

Parameters

The following reports have additional sections for which results can be output.

- area_report – The area report contains results for the following sections:
  - io_port
  - non_io_reg
  - total_io_reg
  - v_ram
  - dsp_used
  - total_luts
- timing_report – The timing report contains results for the following sections:
  - clock_name

– req_freq
– est_freq
– slack

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%)
```

Reporting Parameters Independently

You can also specify some of the parameter reporting separately at the command line. For example:

- Report Timing

```
% report_timing
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 functions 0
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
```

synplify_pro

Starts the FPGA synthesis tool and runs synthesis from the command line. Use the appropriate command for the tool you are using. 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	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.
-Identify_dir *dir*	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 *impName*	Runs only the specified implementation. You can use this option in conjunction with the -batch keyword (the Synplify tool supports only a single implementation).

Option	Description
-ip_license_wait *waitTime*	Specifies how long to wait for a Synopsys DesignWare IP license when one is not immediately available. e. If you do not specify the -ip_license_wait option, license queuing is not enabled. 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. 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 -batch keyword. For details, see Queuing Licenses, on page 461 in the *User Guide*. The *waitTime* value determines license queuing and sets a maximum wait time: • Undefined or 0 = Queuing off • 1 = Queuing enabled, indefinite wait time • >1 = Queuing enabled for the specified time
-license_release	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. This command option must be run in batch mode. Specify the following command: `toolName -batch -license_release`
-licensetype *featureName*	Specifies a license if you work in an environment with multiple Synopsys FPGA licenses. You can use this option in conjunction with the -batch keyword.
-license_wait *waitTime*	Specifies how long to wait for a Synopsys FPGA license. If you do not specify the -license_wait option, license queuing is not enabled. 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 -batch keyword. For details, see Queuing Licenses, on page 461 in the *User Guide*. The *waitTime* value determines license queuing and sets a maximum wait time in seconds: • Undefined or 0 = Queuing off • 1 = Queuing enabled, indefinite wait time • >1 = Queuing enabled for the specified wait time

Option	Description
-log *filename*	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	Starts synthesis tool in shell mode. Note: The FPGA synthesis tools only support the -shell option on UNIX and Linux platforms.
-tcl *prjFile* | *Tclscript*	Starts the synthesis tool in the graphical user interface using the specified project or Tcl file.
-tclcmd *command*	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 86
- Log File Commands, on page 86
- Technology-specific Tcl Commands, on page 86

Synthesis Commands

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

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.

Technology-specific Tcl Commands

You can find vendor-specific Tcl commands in the appropriate vendor chapter.

Vendor/Family	Tcl Commands Described in...
Microsemi	Microsemi Tcl set_option Command Options, on page 710

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

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 106).

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.

The table summarizes where to find detailed information:

For...	**See...**
Command syntax	Tcl Find Syntax, on page 91
Syntax details: object types, expressions, case sensitivity, and special characters	Tcl Find Command Object Types, on page 94 Regular Expressions, Wildcards, and Special Characters, on page 94 Tcl Find Command Case Sensitivity, on page 96
Examples of find syntax	Demos and Examples button, accessible from the tool UI Tcl Find Syntax Examples, on page 97
Filtering find searches by property	Tcl Find -filter Command, on page 100 Find Filter Properties, on page 101 Refining Tcl Find Results with -filter, on page 132 in the *User Guide*.
Using find search patterns and using find in collections	Finding Objects with Tcl find and expand, on page 132 in the *User Guide*.

Tcl Find Syntax

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

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 pattern*	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 94. *pattern* 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 -nocase or -regexp. See Regular Expressions, Wildcards, and Special Characters, on page 94 for additional information.
-filter *expression*	Refines the results of find further, by filtering the results by the specified object property. For details about the syntax, refer to Tcl Find -filter Command, on page 100. If you use the -filter option, it must be specified as the last argument to the find command.
-flat	Allows wildcard * to match the hierarchy separator.

Argument	Description
-hier	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. 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 94.
-hsc *separator*	Specifies the hierarchy delimiter character. The default is the dot (.). The dot can be ambiguous if it is used both as a delimiter and as a normal character. For example, block1.u1 could mean the instance u1 in block1, or a record named block1.u1. Use the -hsc *separator* option to specify an unambiguous character as the hierarchy delimiter. For example, find -hsc "@" [block1@u1] finds the hierarchical instance in the block, while find -hsc "@" [block1.u1] finds the record. See Checking Constraint Files, on page 53 for more information.
-in $*collectionName*| *listName*	Restricts the search to the specified list or collection.
-leaf	Returns only non-hierarchical instances.
-nocase	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 96 for more information.
-namespace techview | netlist	Determines the database to search for the find operation. • techview searches the mapped (srm) database. This is the default. • netlist searches the output netlist. This option is not available for an RTL view.

Argument	Description

-print	Prints the first 20 results. For a full list of objects found, use c_print or c_list.
	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.
	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:
	`"i:reg1"`
	`"i:\weird_name[foo$]"`
	`"i:reg2"`
	`<<found 233 objects. Displaying first 20 objects. Use c_print or c_list for all. >>`

-regexp	Treats the pattern as a regular expression instead of a simple wildcard pattern. It also uses the == and != filter operators to compare regular expressions, rather than simple wildcard patterns. You cannot use this argument with -nocase or -exact. If you do not specify -regexp, there are only two special characters that are used as wildcards: * and ?. See Regular Expressions, Wildcards, and Special Characters, on page 94 for details about regular expressions.
-rtl | -tech	Uses the most recently activated RTL or Technology view. If none are available, it opens a new view. The RTL view is the default.

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.rt1|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.

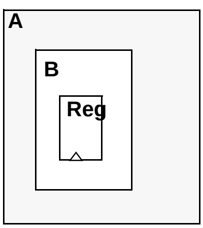
Syntax	Matches...
Meta Characters: Used to match certain conditions in a string	
^	At the beginning of the string
$	At the end of the string. Use curly brackets {} or double quotes to prevent the interpretation of special characters within a pattern.
.	Any character. If you want to use the dot (.) as 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 *k* 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 *c* 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	
[*list*]	Any single character from the *list*. 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.
[^*list*]	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

Syntax	Matches...
\w	A word character; i.e., alphanumeric characters or underscores
\W	A non-word character
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*<=*P*

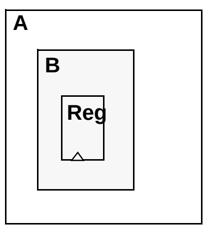
Tcl Find Command Case Sensitivity

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.

Verilog
VHDL



i:A.B.Reg - correct
i:A.b.Reg - correct
i:a.B.Reg - correct
i:a.b.Reg - correct
i:A.B.REG - incorrect
i:A.B.reg - incorrect



i:A.B.Reg - correct
i:A.b.Reg - incorrect
i:a.B.Reg - incorrect
i:a.b.Reg - incorrect
i:A.B.REG - correct
i:A.B.reg - correct

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.
find -hier-seq {*} -filter @slack <{-0.0}	Finds all sequential elements with negative slack.
find -hier-seq {*} -filter {@clock ==clk1}	Finds all sequential elements within the clk1 clock domain
find -hier-net {*} -filter {@fanout >20}	Finds high fanout nets that drive more than 20.destinations
find -hier-seq * -in $all_inst_coll	Finds sequential elements inside the all_inst_coll collection
find -net -regexp {[a-b].*}	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
```

Tcl Find -filter Command

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 100
- Find Filter Properties, on page 101
- Find Filter Examples, on page 104

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. You can use the following operators: • Relational operators: =, <, >, ==, >=, <= • Logical operators: &&, ||, !
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.

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	Comment
Common Properties			
type	view | port | net | instance | pin]	All	
View Properties			
compile_point	locked	Tech	
is_black_box	1	All	
is_verilog	0 | 1	All	
is_vhdl	0 | 1	All	
syn_hier	remove | flatten | soft | firm | hard	Tech	
Port Properties			
direction	input | output | inout	All	
fanout	*value*	All	Total fanout (integer)
Instance Properties			
area	*area_value*	Tech	
arrival_time	*value*	Tech	Corresponds to worst slack

Property Name	Property Value	HDL View	Comment
async_reset	**n:***netName*	All	
async_set	**n:***netName*	All	
clock	*clockName*	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:***netName*	All	Highest branch name in the hierarchy, and closest to the driver
compile_point	locked	Tech	Automatically inherited from its view
hier_rtl_name	*hierInstanceName*	All	
inout_pin_count	value	All	
input_pin_count	value	All	
inst_of	viewName	All	
is_black_box	1 (Property added)	All	Automatically inherited from its view
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

Property Name	Property Value	HDL View	Comment
kind	*primitive* For example: inv | and |dff | mux | statemachine | ...)	All	Tech view contains vendor-specific primitives
location	(*x*, *y*)	Tech	Format can differ
name	*instanceName*	All	
orientation	N | S | E | W	Tech	
output_pin_count	*value*	All	
pin_count	*value*	All	
placement_type	unplaced | placed	All	
rtl_name	*nonhierInstanceName*	All	
slack	*value*	Tech	Worst slack of all arcs
slow	1	Tech	
sync_reset	**n:***netName*	All	
sync_set	**n:***netName*	All	
syn_hier	remove | flatten | soft | firm | hard	Tech	Automatically inherited from its view
view	*viewName*	All	
Pin Properties			
arrival_time	*timingValue*	Tech	
clock	*clockName*	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	*value*	All	Total fanout (integer)
is_clock	0 | 1	All	

Property Name	Property Value	HDL View	Comment
is_gated_clock	0|1	All	Set in addition to is_clock
slack	*value*	Tech	
Net Properties			
clock	*clockName*	All	Could be a list if there are multiple clocks
is_clock	0|1	All	
is_gated_clock	0|1	All	Set in addition to is_clock
fanout	*value*	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:

`c_print [find -hier -view{ *} -filter (@is_black_box)]`	Finds all objects that are black boxes.
`c_print [find -hier -view {*} -filter (!@is_black_box)]`	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 * -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 136 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 *object*	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	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.
-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.
-*objectType*	Optionally specifies the type of object to be returned by the expansion. If you do not specify an *objectType*, all objects are returned. The object type is one of the following: • **-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	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.
	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 double quotes 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. >>`

-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.
-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 all sequential elements.

Tcl expand Syntax Examples

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

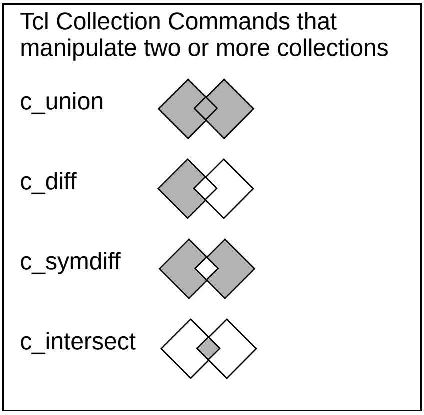
Collection Commands

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 173) or in a Tcl file.

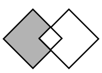
The following table lists the commands for creating, copying, evaluating, traversing, and filtering collections, and subsequent sections describe the collections, except for find and expand, in alphabetical order. For information on using collections, see Using Collections, on page 140 in the *User Guide*.

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}
```

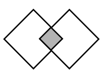
c_info

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 $intercollection
   {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:

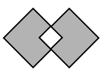
```
c_print -file foo.txt $col
```

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 111).

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.

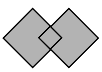
```
set symDiff_collection [c_symdiff $collection1 $collection2]
```

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
```

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*

collectionName	The name of the new collection.
collectionCriteria	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 the these commands for examples.
copyName	The name assigned to the copied collection.
$*collectionName*	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 110, c_intersect, on page 111c_symdiff, on page 113, c_union, on page 114, and define_collection, on page 115.

Object 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 90 and expand, on page 106).

The Synopsys SDC collection commands are only intended to be used in the FDC file to create collections of objects for constraints. This section describes the syntax for the object query commands supported in the FPGA synthesis tools. For complete documentation on these commands, refer to the Design Compiler documentation.

- all_clocks
- all_inputs
- all_outputs
- all_registers
- get_cells
- get_clocks
- get_nets
- get_pins
- get_ports

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
```

Object 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 Object Query Commands and Tcl find and expand Commands, on page 118 for examples.

Object 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: set_multicycle_path {4} -from [all_registers -no_hierarchy -output_pins -clock [get_clocks {clkb}]] set_multicycle_path {4} -from [get_pins -of_objects [get_cells * -filter {@clock == clkb}] -filter {@name == Q}]
find	Tcl Window: % define_collection [regsub -all {i:([^\s]+)} [join [c_list [find -inst * -filter @clock == clkfx]]] {t:\1.Q}]
Return all registers in the design clocked by the rising edge of clock clkfx	
all_registers	FDC Constraint: 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}]
find	Tcl Window: find -hier -inst * -filter {@clock == clkfx && @clock_edge == rise}
Return clock pins of all registers clocked by the falling edge of cklfx	
all_registers	FDC Constraint: 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}]
find	Tcl Window: % find -hier -inst * -filter {@clock == clkfx && @clock_edge == fall}
Return the E pins of all instances of dffre cells (e.g. inst1.inst2.my_reg.E)	
get_pins	FDC Constraint: set_multicycle_path -to [get_pins -filter {@name == E} -of_objects [get_cells -hier * -filter {@inst_of == dffre}]
find	Tcl Window and FDC Constraint: % regsub -all {i:([^\s]+)} [join [c_list [find -hier -inst * -filter @inst_of == dffre]]] {t:\1.E}]

all_clocks

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_inputs

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

Example

The following constraint sets a default input delay.

```
set_input_delay 3 [all_inputs]
```

all_outputs

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

Example

The following constraint sets a default output delay.

```
set_output_delay 2 [all_outputs]
```

all_registers

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

Example

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

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

get_cells

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
```

get_clocks

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*]

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 `-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 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.
pattern	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 *objects*	Creates a collection of clocks that are defined for the given net or pin objects.

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_nets

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 `block1/muxsel` a hierarchical search uses `muxsel` to find this net 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 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.

pattern	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. 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.
-of_objects *objects*	Creates a collection of nets connected to the specified objects. Each object can be a pin, port, or cell.

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

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 *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. The `-regexp` and `-exact` options are mutually exclusive; use only one.
-exact	Treats wildcards as plain characters, so the meanings of these wildcard are not interpreted. The `-regexp` and `-exact` options are mutually exclusive; use only one.
-filter *expressions*	Filters the collection with the specified expression. 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.
pattern	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. 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.

-of_objects *objects*	Creates a collection of pins connected to the specified objects. Each object can be a cell or net. 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. You cannot use the `-hierarchical` option with the `-of_objects` option.
-leaf	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.

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, and 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}]
```

get_ports

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	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 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.
pattern	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. 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.

Examples

The following example queries all input ports beginning with mode.

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

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

collection1	Specifies the base collection to which objects are to be added. This collection is copied to a resulting collection, where objects matching *objectSpec* are added to this results collection.
objectSpec	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 131 and Homogeneous Base Collection, on page 132.

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 Also

- 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

variableName	Specifies a variable name. The objects matching *objectSpec* are added to the collection referenced by this variable.
objectSpec	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 (append_to_collection, on page 133) 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 Also

- 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
[*collection*1]

Arguments

collection1	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

iterationVariable	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 *iterationVariable*, as they are the same data type.
collections	Specifies a list of collections on which to iterate.
body	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 foreach_in_collection 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

collection1	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

collection1	Specifies the collection to be searched.
index	Specifies an index to the collection. Allowed values are integers from 0 to sizeof_collection -1.

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 Also

- 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 *not* found in *objectSpec*. By default, when this option is not specified, objects are removed from the base collection that are found in the *objectSpec*.
collection1	Specifies the base collection that is copied to a resulting collection, where objects matching *objectSpec* are removed from this results collection.
objectSpec	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 Also

- 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

collection1	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 commands and ways to access them in the graphical user interface (GUI).

File Menu

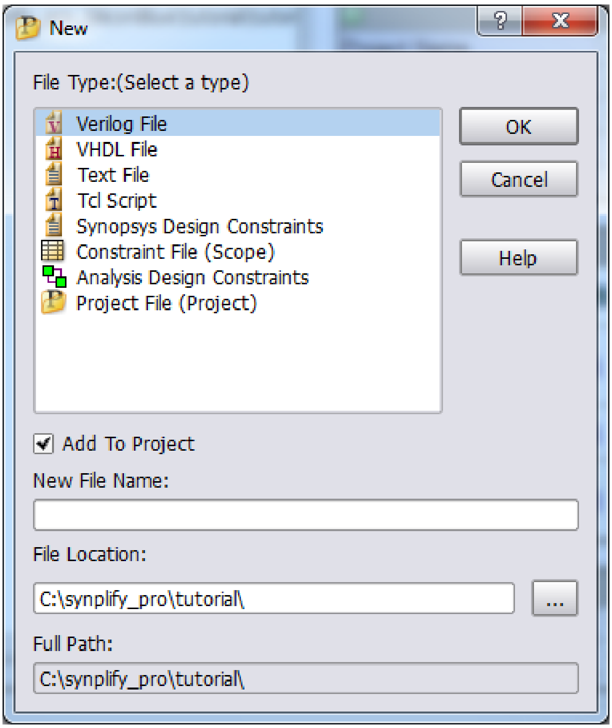
Use the File menu for opening, creating, saving, and closing projects and files. The following table describes the File menu commands.

Command	Description
New	Creates one of the following types of files: Text, Tcl Script, VHDL, Verilog, Synopsys Design Constraints, Constraint, Analysis Design Constraint, or Project. See New Command, on page 147.
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	This command is available in the following views: • HDL Analyst Views • FSM Viewer 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 148.
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 150.
Open Project	Opens a project. See Open Project Command, on page 150.
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 280.
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 six files you opened 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 (Verilog, VHDL, text, Tcl script, Synopsys design constraints, constraint, analysis design constraints, 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.



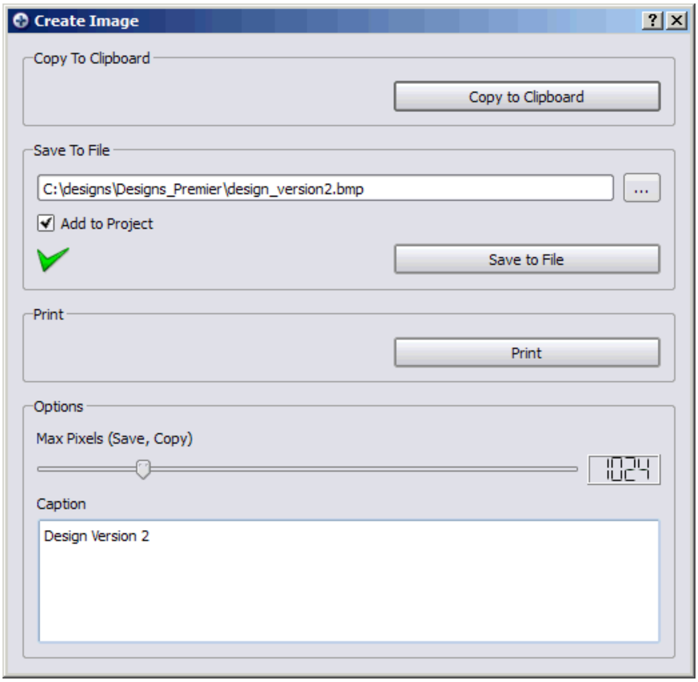
File Type	Opens Window	Directory Name	Extension
Verilog	Text Editor	Verilog	.v
VHDL	Text Editor	VHDL	.vhd
Text	Text Editor	Other	.txt
Tcl Script	Text Editor	Tcl Script	.tcl
FPGA Design Constraints	SCOPE	Constraint	.fdc
Constraint	SCOPE	Constraint	.sdc
Analysis Design Constraints	SCOPE	Analysis Design 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 caption is overlayed at the top-left corner of the image.

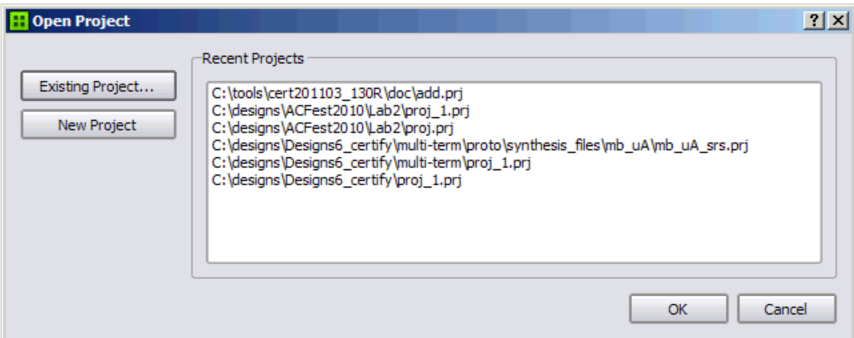
Build Project Command

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 171.

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:

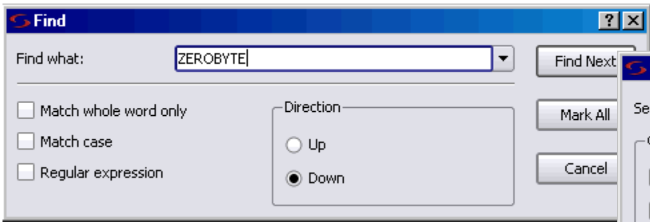
Command	Description
Basic Edit Menu Commands	
Undo	Cancels the last action.
Redo	Performs the action undone by Undo.
Cut	Removes the selected text and makes it available to Paste.
Copy	Duplicates the selected text and makes it available to Paste.
Paste	Pastes text that was cut (Cut) or copied (Copy).
Delete	Deletes the selected text.
Find	Searches the file for text matching a given search string; see Find Command (Text), on page 153. 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 155. In the project view, searches files for text strings; see Find Command (In Project), on page 154.

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 159.
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 160.
Goto	Goes to a specific line number. See Goto Command, on page 161.
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).

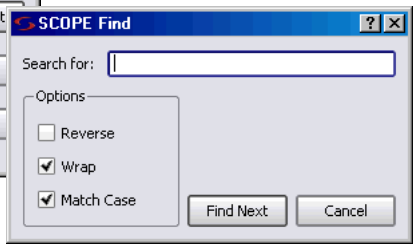
Find Command (Text)

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 154, to search for files in the Project.

The HDL Analyst Find command is different; see Find Command (HDL Analyst), on page 155 for details.



In Text Editor



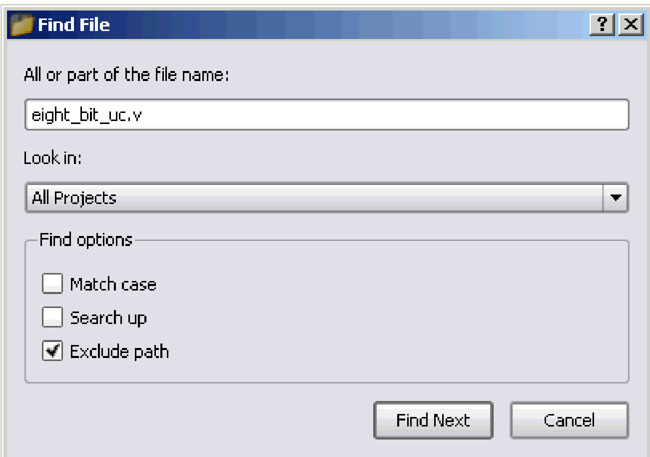
In SCOPE

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.

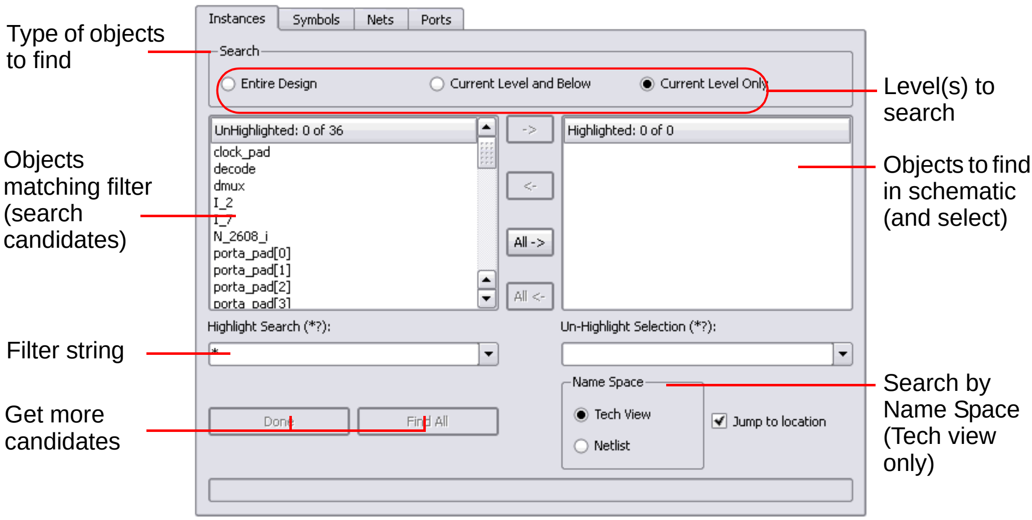


Field/Option	Description
All or part of the file name	Search string matching the file to find. You can specify all or part of the file name.
Look in	Search for files in all projects or limit the search to files only in the specified project.
Match Case	When enabled, searching is case sensitive.

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 228 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 228 of the *User Guide*.
UnHighlighted	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. 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. 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. To confirm a selection for highlighting and to move the selected objects to the Highlighted list, click the -> button.

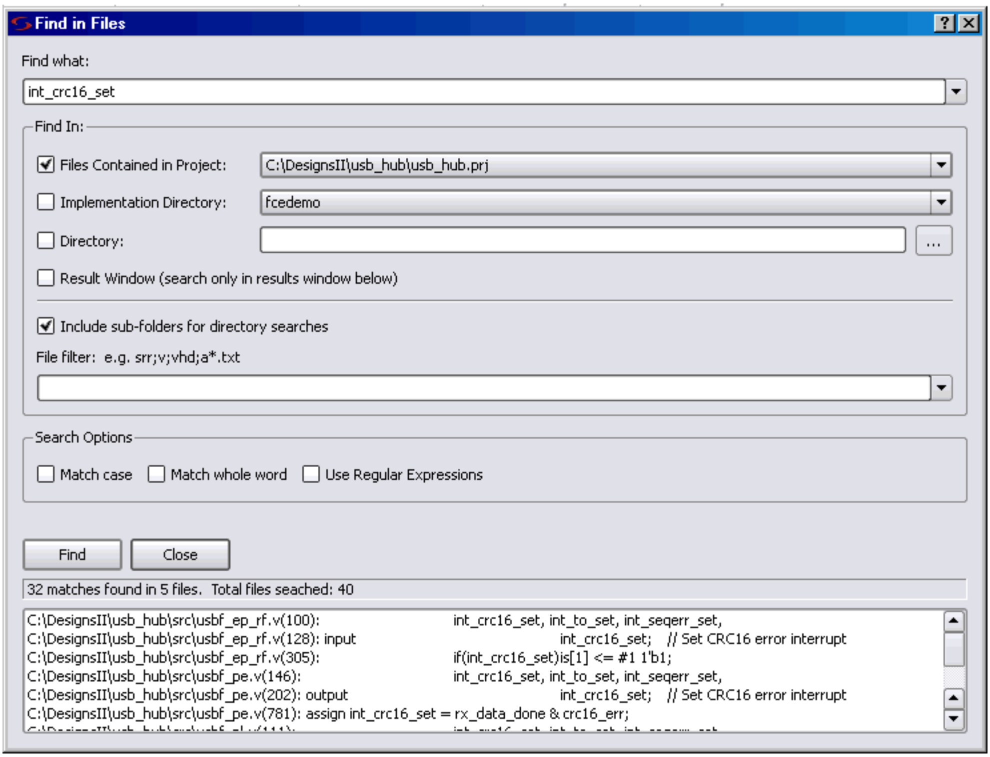
Field/Option	Description
Highlight Search (*?)	Determines which object names appear in the UnHighlighted area, based on the case-sensitive filter string that you enter. For tips about using this field, see Using Wildcards with the Find Command, on page 231 of the *User Guide.* The filter string can contain the following wildcard characters: • * (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. 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). 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. 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.
->	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.
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.

Field/Option	Description
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 (srm) database. For more information, see Using Find for Hierarchical and Restricted Searches, on page 228 of the *User Guide*.
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 228 of the *User Guide*.
Find 200	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. Using the Enter (Return) key when the cursor is in the Highlight Search (*?) field is equivalent to clicking the Find 200 button. *Usage note:* Click Find 200 before Find All to prevent unwanted matches in case the Highlight Search (*?) string is less selective than you expect.
Find All	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 *Usage Note* for Find 200, above.

For more information on using the Object Query dialog box, see Using Find for Hierarchical and Restricted Searches, on page 228 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 95 of the *User Guide*.

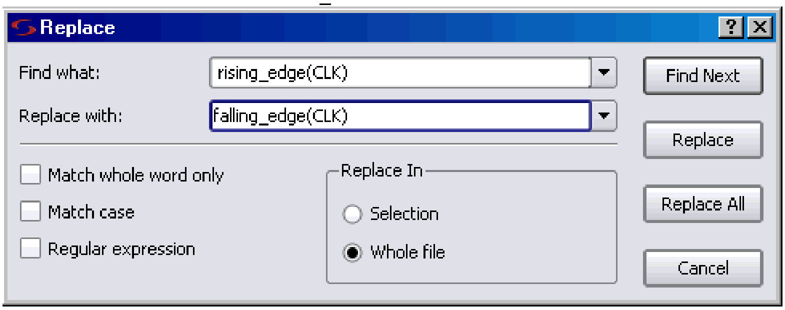


Field/Option	Description
Find what:	Text string object of search.
Files Contained in Project	Drop-down menu identifying the source project of the files to be searched.
Implementation Directory	Drop-down menu restricting project search to a specific implementation or all implementations.
Directory	Identifies 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 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.

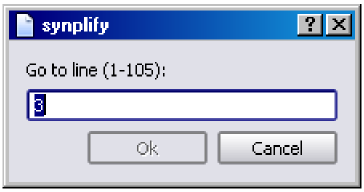


Feature	Description
Find what	Search string matching the text to find. You can use the pull-down list to view and reuse search strings used previously in the current session.
Replace with	The text that replaces the found text. You can use the pull-down list to view and reuse replacement text used previously 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).

Feature	Description
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 162
- View Menu: Zoom Commands, on page 163
- View Menu: RTL and Technology Views Commands, on page 163
- View Menu: FSM Viewer Commands, on page 164

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: • 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 165.
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 167.
View Result File	Displays a detailed netlist report.

View Menu: Zoom Commands

Command	Description
Zoom In Zoom Out	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.
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 162 and View Menu: Zoom Commands, on page 163.

Command	Description
Push/Pop Hierarchy	Traverses design hierarchy using the push/pop mode – see Exploring Design Hierarchy, on page 218 of the *User Guide*.
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 166.
Visual Properties	Toggles the display of information for nets, instances, pins, and ports in the HDL Analyst view. To customize the information that displays, set the values with Options->HDL Analyst Options->Visual Properties. See Visual Properties Panel, on page 294.
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 162 and View Menu: Zoom Commands, on page 163.

.

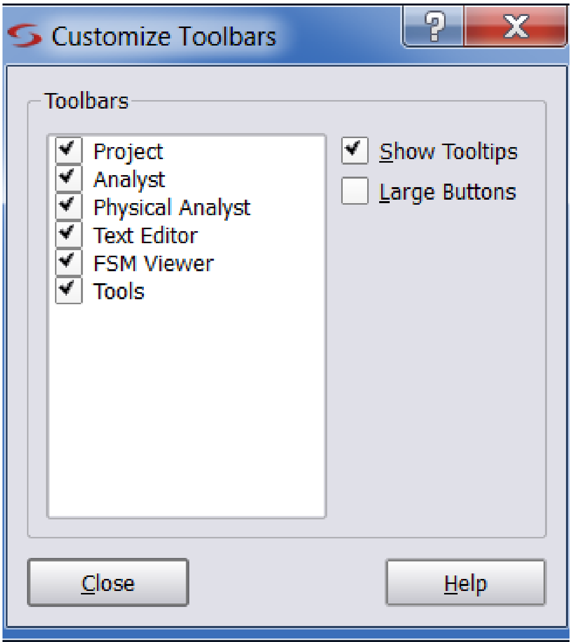
Command	Description
Filter->Selected	Hides all but the selected state(s).
Filter->By output transitions	Hides all but the selected state(s), their output transitions, and the destination states of those transitions.
Filter->By input transitions	Hides all but the selected state(s), their input transitions, 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

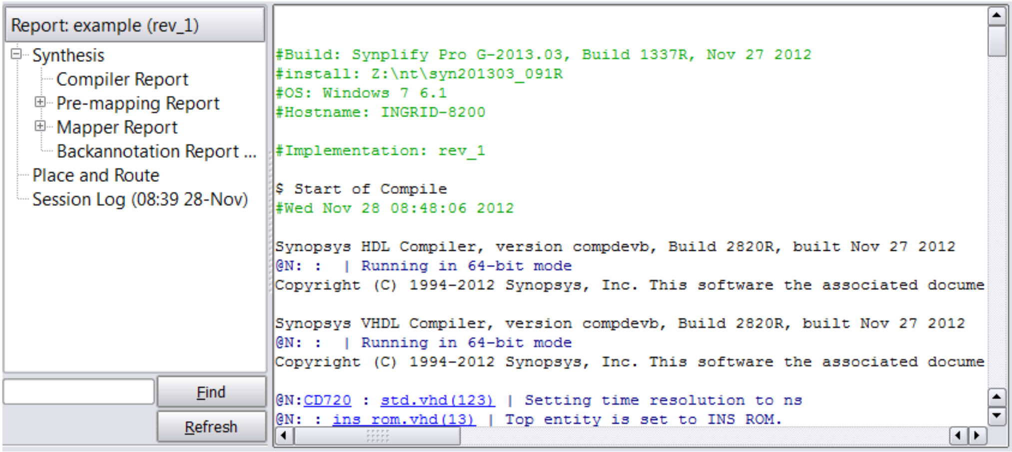
Select View->View Sheets to display the Goto Sheet dialog box and select a sheet to display. The Goto Sheet dialog box is only available in an RTL or Technology view, and only when a multiple-sheet design is present.

To see if your design has multiple sheets, check the sheet count display at the top of the schematic window.

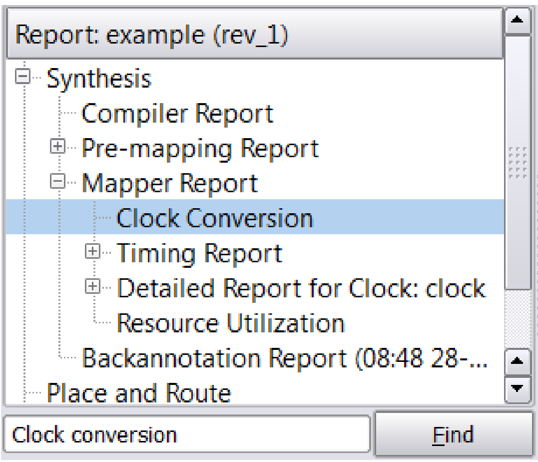
View Log File Command

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.

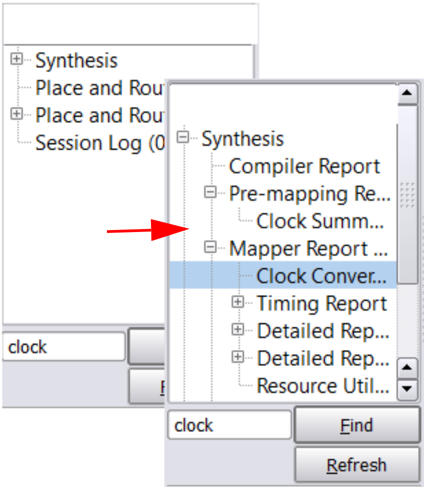
When opening the log file, a table of contents appears. Selecting an item from the table of contents takes you to the corresponding HTML page. To go back, right-click the HTML page and select Back from the menu.



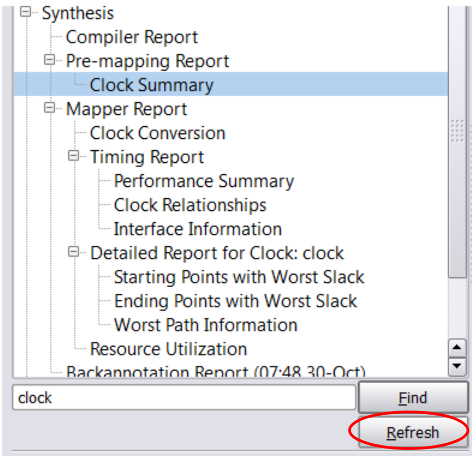
You can use the search field to find an item in the table of contents. Enter all or part of the header name in the search field, then click Find. The log file displays the resulting section.



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 Synplify Pro tools provide 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.

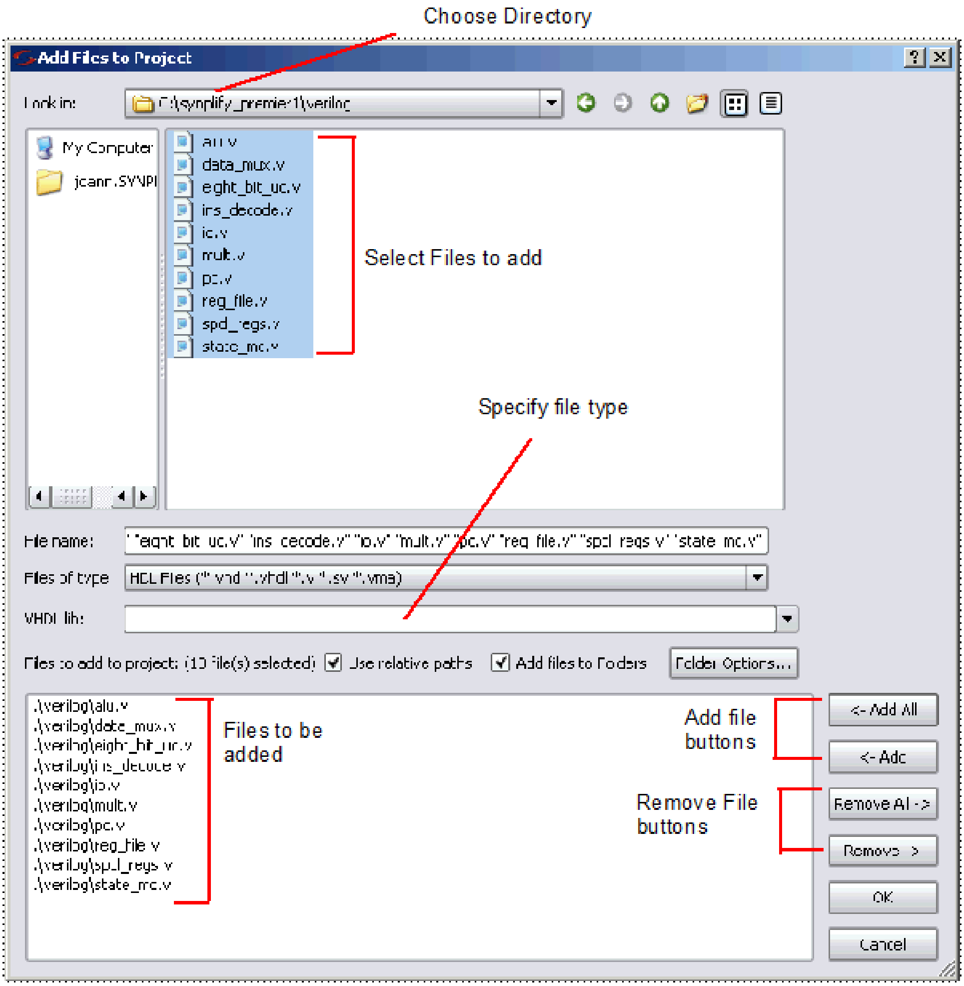
The following table describes the Project menu commands.

Command	Description
Implementation Options	Displays the Implementation Options dialog box, where you set options for implementing your design. See Implementation Options Command, on page 183.
Add Source File	Displays the Select Files to Add to Project dialog box. See Add Source File Command, on page 171. Tcl equivalent: **add_file -fileType** *filename*
Remove File From Project	Removes selected files from your project. Tcl equivalent: **project_file -remove** *filename*
Change File	Replaces the selected file in your project with another that you choose. See Change File Command, on page 173. Tcl equivalent: **project_file -name "***originalFile***" "***newFile***"**
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 174.
Add Implementation	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 174). Tcl equivalent: **impl -add** *implementation_1 implementation* **-type** *implementationType*

Command	**Description**
New Identify Implementation	Creates a new Identify implementation for a current design. To launch the Identify tool set, see the Launch Identify Instrumentor Command, on page 211 and Launch Identify Debugger Command, on page 212. Tcl equivalent: **impl -add** *implementation_1 implementation*
Convert Vendor Constraints	Not applicable for Microsemi technologies.
Archive Project	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 175 for a description of the utility wizard options.
Un-Archive Project	Loads an archived project file to the specified directory. See Un-Archive Project Command, on page 177 for a description of the utility wizard options.
Copy Project	Creates a copy of a design project. Use this command to create a copy of a full or partial project. See Copy Project Command, on page 180 for a description of the utility wizard options.
Hierarchical Project Options	Not applicable for Microsemi technologies.
Add SubProject Implementation	Not applicable for Microsemi technologies.

Add Source File Command

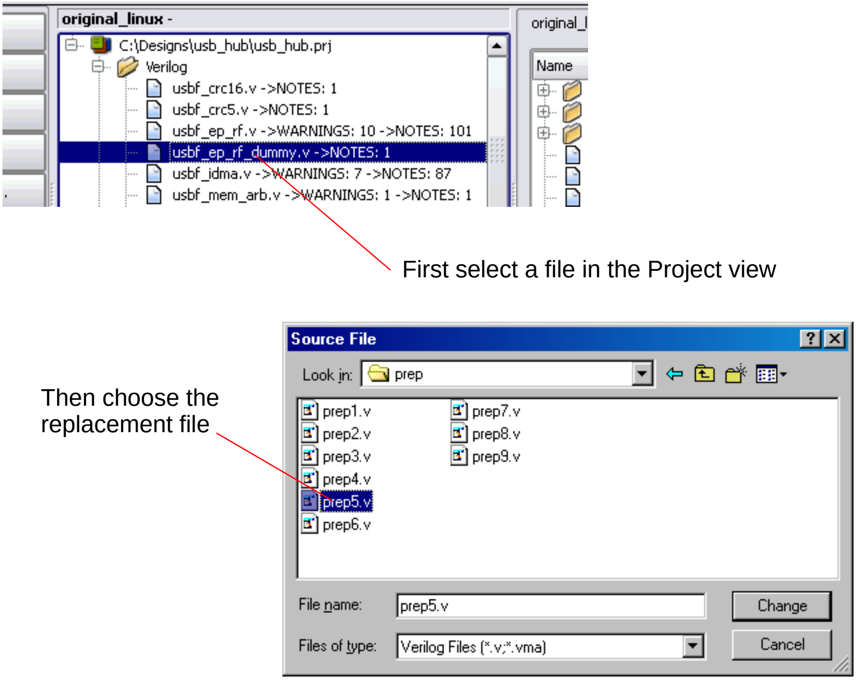
Select Project->Add Source File to add files, such as HDL source files, to your project. This selection displays the Select Files to Add to Project dialog box.


Choose Directory
Add Files to Project
Look in:
My Computer
Select Files to add
Specify file type
File name:
Files of type
VHDL lib:
Use relative paths
Add files to Folders
Folder Options...
Files to be added
Add file buttons
Remove File buttons
<- Add All
<- Add
Remove All ->
Remove ->
OK
Cancel

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.
Files To Add To Project	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. For information about adding files to custom folders, see Creating Custom Folders, on page 64. Tcl equivalent: **add_file -type** *filename*
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	When you add files to folders, you can specify the folder name as either the: • Operating System (OS) folder name • Parent path name from a list provided in the display

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.



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 321. 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 183. 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.

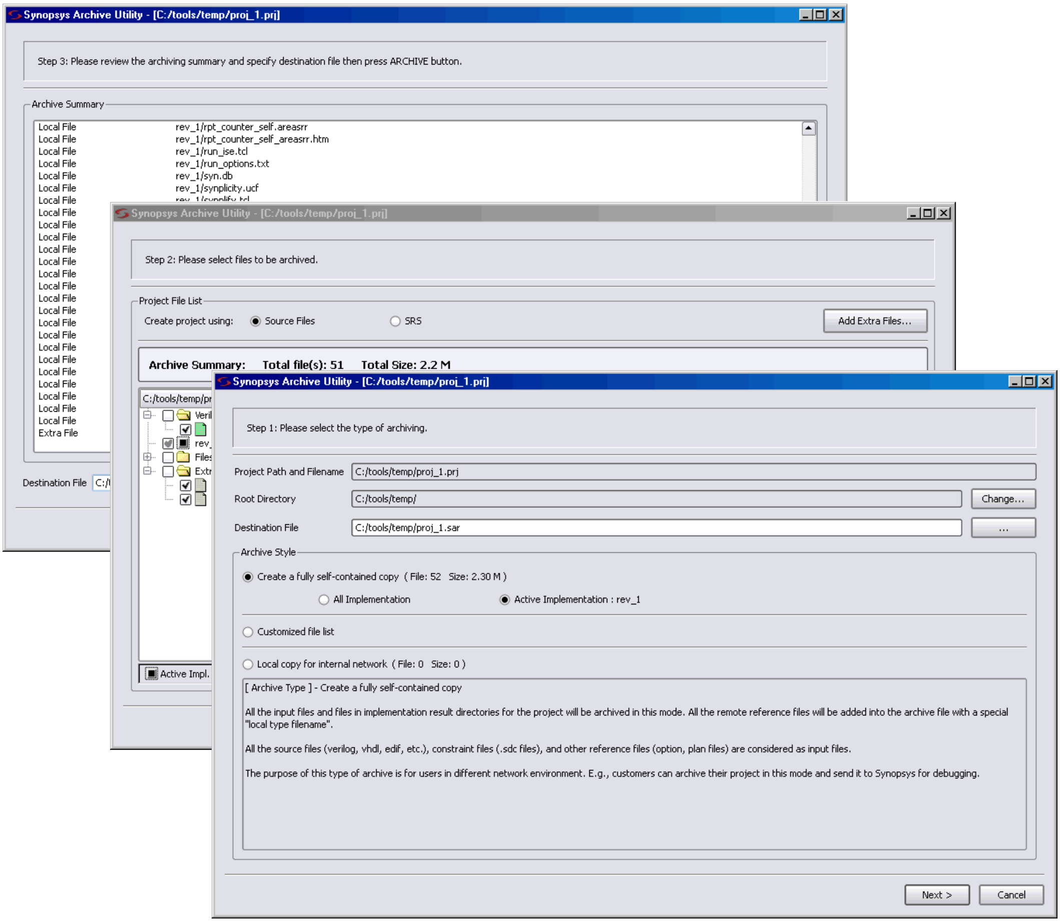
Convert Vendor Constraints Command

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.

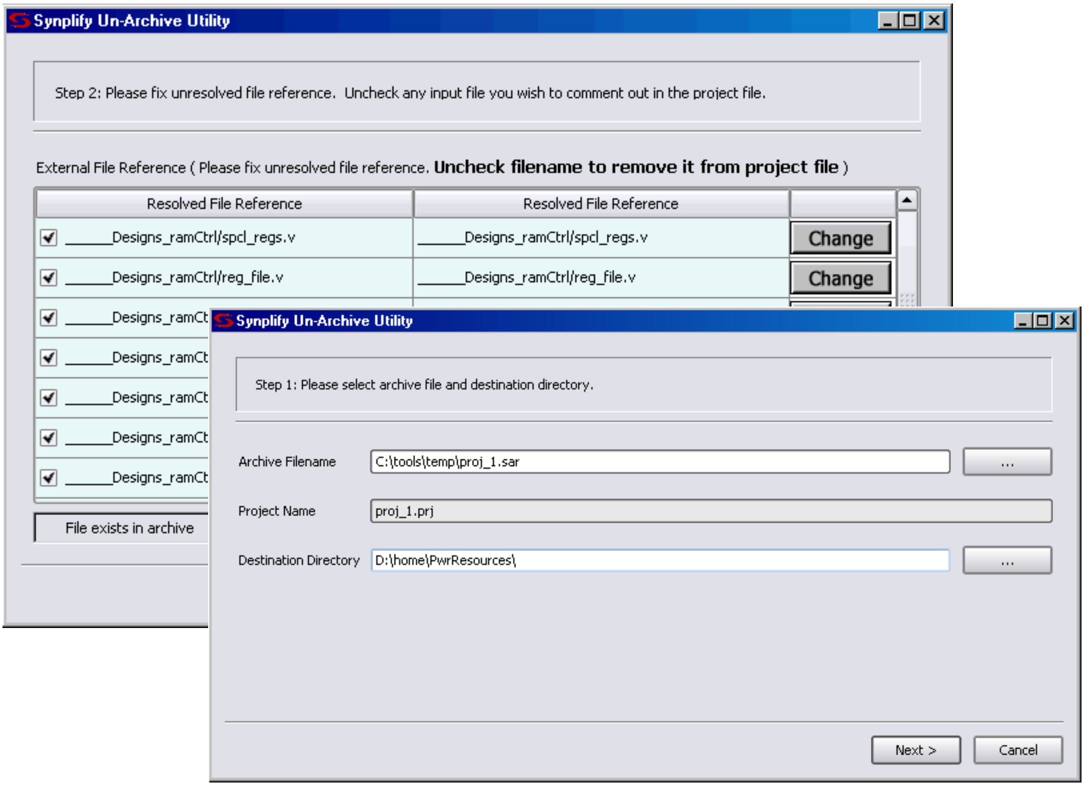
Option	Description
Archive Style	The type of archive: • Create a fully self-contained copy – all project files are archived; includes project input files and result files. • If the project contains more than one implementation: - All Implementation includes all implementations in the project. - Active 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	If you select the Customized file list option in the wizard, you can choose one the following options on the second tab: • 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	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.

For step-by-step details on how to use the archive utility, see Archive Project Command, on page 175.

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.

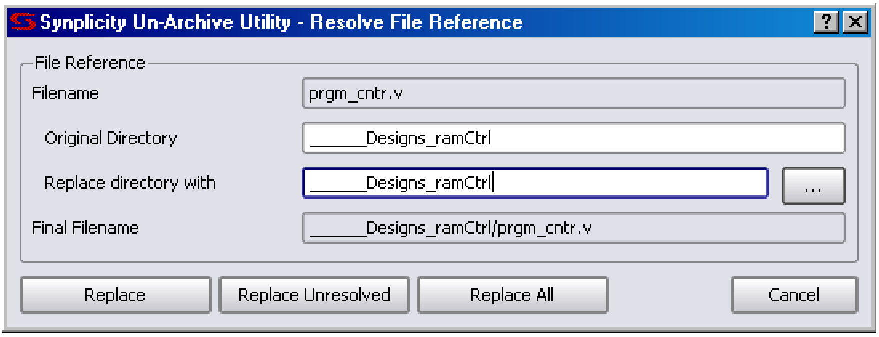


Option	Description
Archive Filename	Path and filename of the .prj file.
Project Name	Top-level directory that contains the project files.
Destination Directory	Pathname of the directory to store the archive .sar file.
Original File Reference/ Resolved File Reference	Displays the files in the archive 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. 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 the change when project files are extracted, use the Change button and select files, as appropriate. See *Resolve File Reference*, next for more details.

For step-by-step details on how to use the un-archive utility, see Un-Archive Project Command, on page 177.

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:



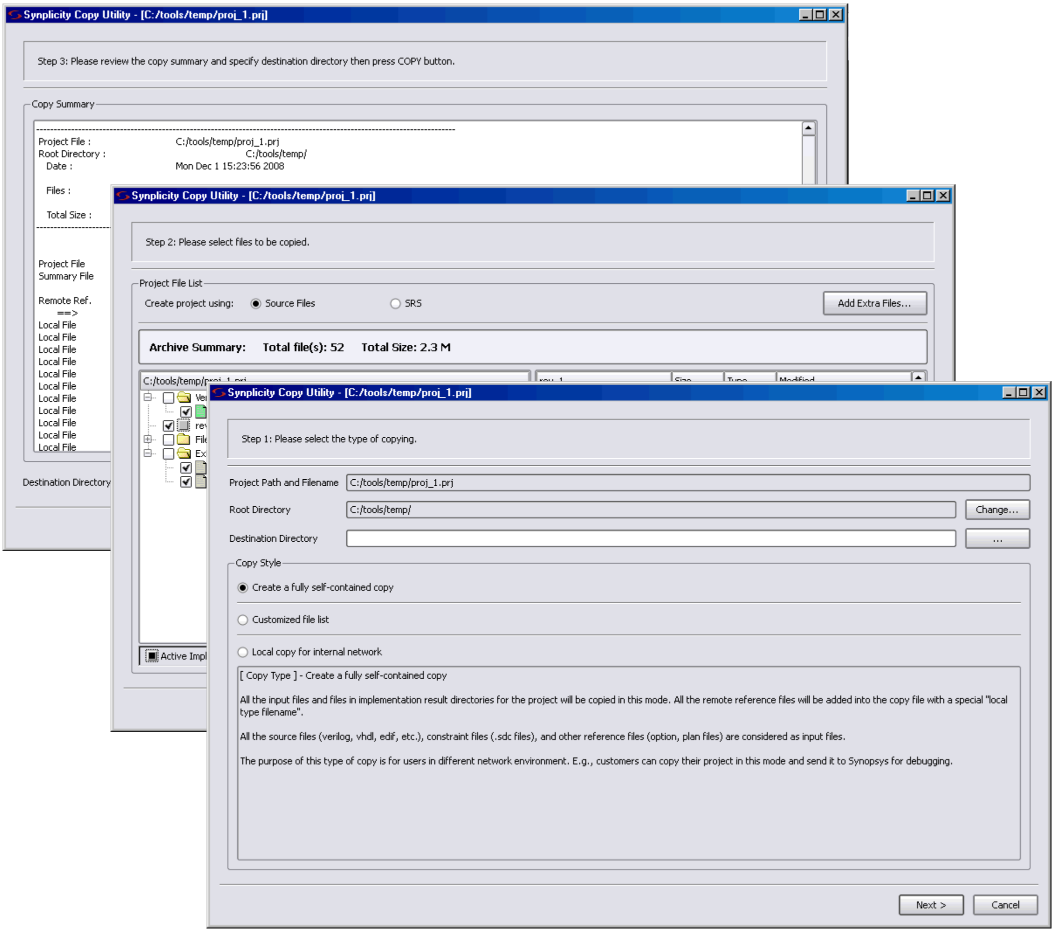
Option	Description
Filename	Specifies the path and name of the file you want to change or resolve.
Original Directory	Specifies the location of the project at the time it was archived.

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	• 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	The type of archive: • 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: - 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	If you select the Customized file list option in the wizard, you can choose one the following options on the second tab: • 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	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.

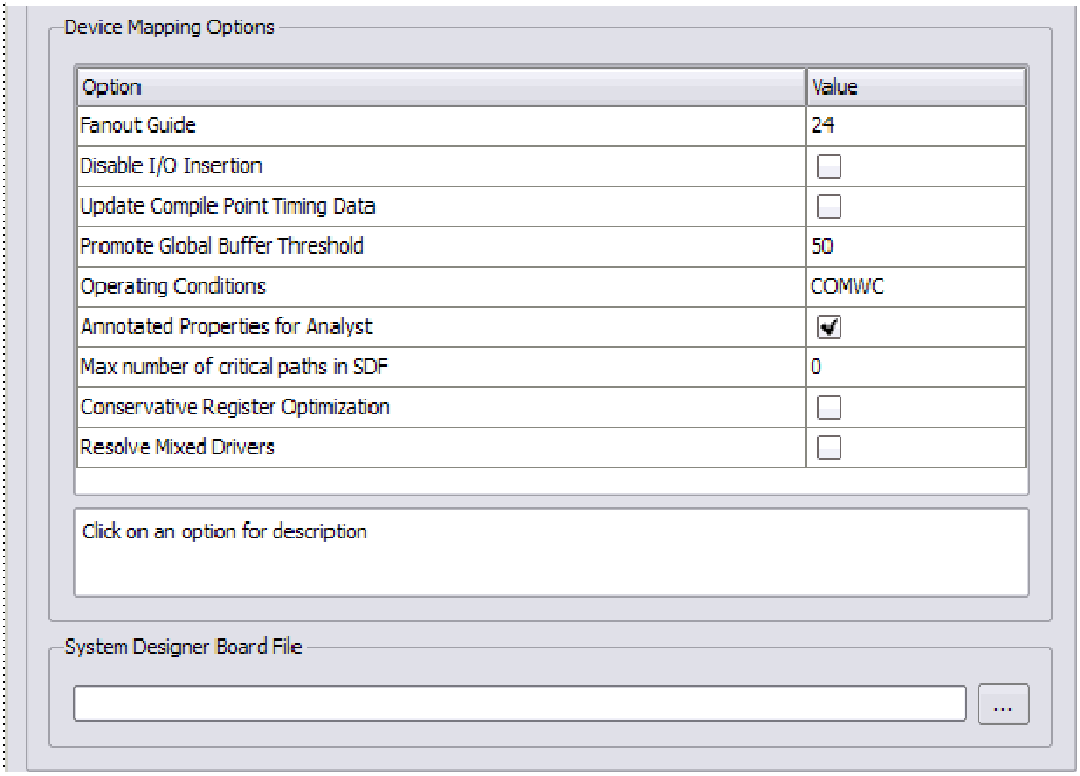
For step-by-step details on how to use the copy utility, see Copy Project Command, on page 180.

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.



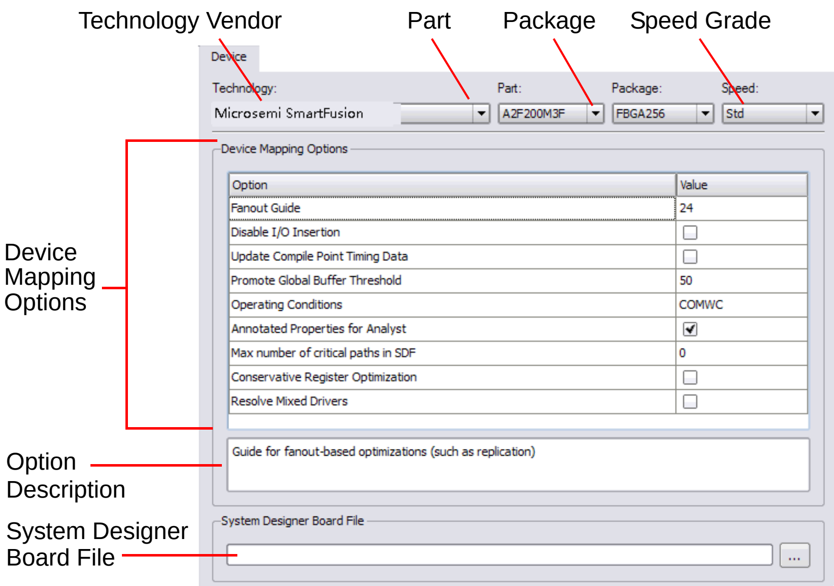
This section describes the following:

- Device Panel, on page 184. For device-specific details of the options, refer to the appropriate vendor chapter.
- Options Panel, on page 185
- Constraints Panel, on page 187
- Implementation Results Panel, on page 189

- Timing Report Panel, on page 191
- VHDL Panel, on page 192
- Verilog Panel, on page 195
- Place and Route Panel, on page 204

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 73 in the *User Guide* for a procedure, and the relevant vendor sections in this reference manual for technology-specific descriptions of the options.

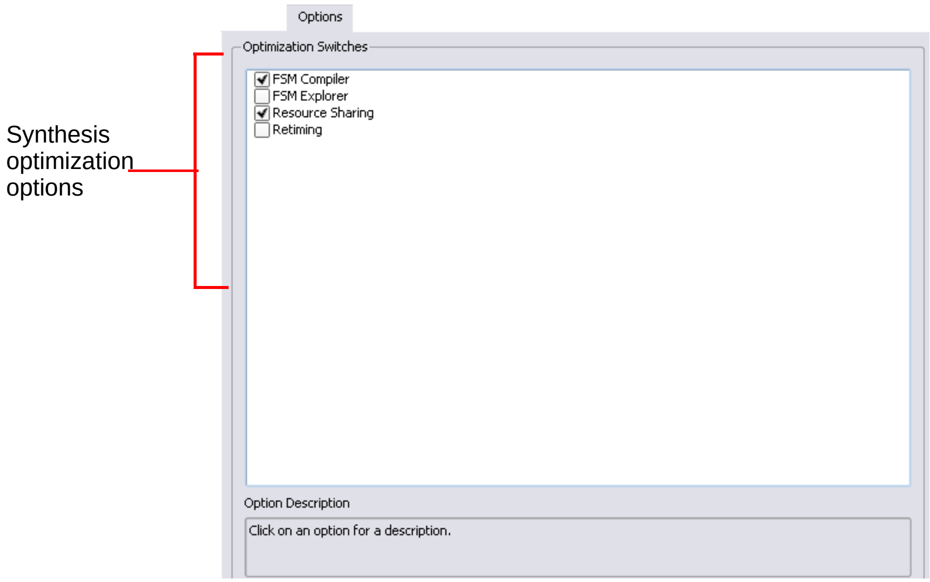
The table below lists the following category of options. Not all options are available for all tools and technologies.

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 *Reference* 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 *Reference* manual.
Option Description	Click on a device mapping option to display its description in this field. Refer to the relevant vendor sections for technology-specific descriptions of the options.
System Designer Board File	Not applicable for Microsemi technologies.

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 76 of the *User Guide* for details.

*



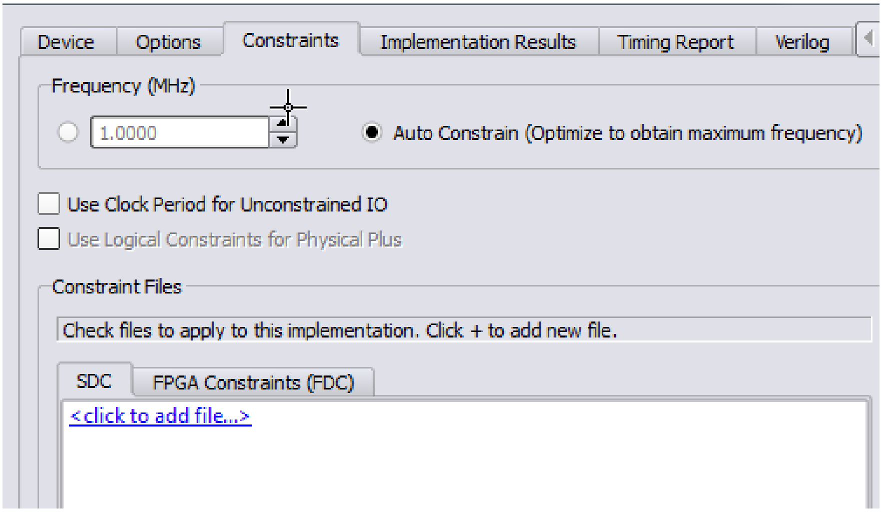
The following table lists the options alphabetically. Not all options are available for all tools and technologies.

Option	Description
Enable 64-bit Synthesis	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. For batch mode, use this Tcl command in your project file: set_option -enable64bit 1 This option is supported on the Windows and Linux platforms. Tcl equivalent: **set_option -enable64bit 0|1**
FSM Compiler	Determines whether the FSM Compiler is run. See FSM Compiler, on page 76 and Optimizing State Machines, on page 347 in the *User Guide.* 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 353 in the *User Guide*. Tcl equivalent: **set_option -use_fsm_explorer 0|1**

Option	Description
HDL Analyst Quick Load (*Beta*)	When enabled, allows the compiler to pre-partition the design into multiple modules that are written to separate netlist files (`srs`). This feature improves memory usage in the HDL Analyst for large designs. By default, this option is disabled. For more information, see HDL Analyst Quick Load, on page 77. Tcl equivalent: **set_option -hdl_qload 0|1**
Resource Sharing	Determines whether you optimize area by sharing resources. When enabled, this optimization technique runs during the compilation stage of synthesis. 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 345 for information for how to use this option in the *User Guide*. Enabling this option generates the resource sharing report in the log file (see Resource Usage Report, on page 261). Tcl equivalent: **set_option -resource_sharing 0|1**
Retiming	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 and DSPs regardless of the Retiming setting. For details about using this feature, see Retiming, on page 328 in the *User Guide*. Tcl equivalent: **set_option -retiming 0|1**

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 78, in the *User Guide* for details.



Option	Description
Frequency	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. Tcl equivalent: **set_option -frequency** *frequency*
Auto Constrain	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 287 in the *User Guide* for information about using this option. You can also set this option in the Project view. Tcl equivalent: **set_option -frequency auto**

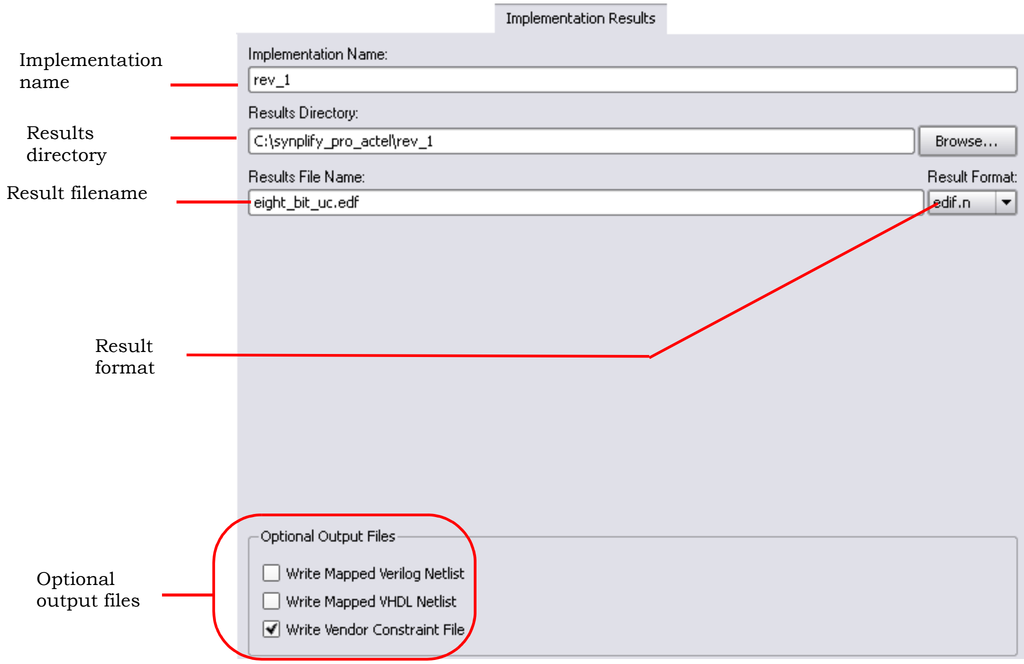
Option	Description
Use clock period for unconstrained IO	Determines whether default constraints are used for I/O ports that do not have user-defined constraints. When disabled, only set_input_delay or set_output_delay constraints are considered during synthesis or forward-annotated after synthesis. When enabled, the software considers any explicit set_input_delay or set_output_delay constraints, as before. 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. Tcl equivalent: **set_option -auto_constrain_io 0|1**
Constraint Files SDC	Specifies which timing constraints (SDC) files to use for the implementation. Select the check box to select a file. For Synplify Pro block-level files in the compile-point flows, the Module column shows the name of the module or compile point.
Constraint Files FDC	Specifies which timing constraints (FDC) files to use for the implementation. Select the check box to select a file. For Synplify Pro, block-level files in the compile-point flows, the Module column shows the name of the module or compile point.

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 80 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.



Option	Description
Implementation Name Results Directory Result Base Name	Displays implementation name, directory path for results, and the base name for the result files. Tcl equivalent: **set_option -result_file** *pathtoResultFile*
Result Format	Select the output that corresponds to the technology you are using. See Generating Vendor-Specific Output, on page 488 in the *User Guide* for a list of netlist formats.
Write Mapped Verilog Netlist Write Mapped VHDL Netlist	Generates mapped Verilog or VHDL netlist files.
Write Vendor Constraint File	Generates a vendor-specific constraint file for forward annotation.

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 263.

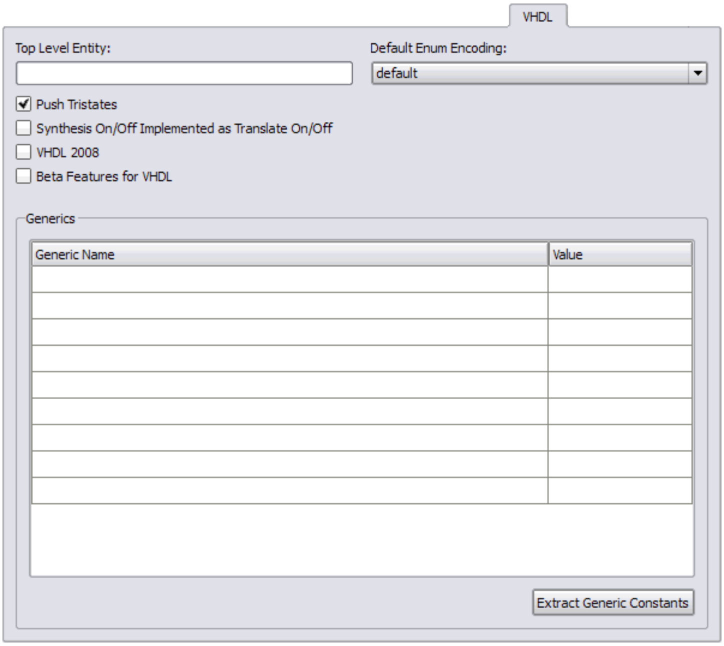
Option	Description
Number of Critical Paths	Set the number of critical paths for the software to report. Tcl equivalent: **set_option -num_critical_paths** *numberOfPaths*
Number of Start/End Points	Specify the number of start and end points to see reported in the critical path sections. Tcl equivalent: **set_option -num_startend_points** *numberOfPoints*

See also:

- Timing Reports, on page 263, for more information on the default timing report, which is affected by the Timing Report panel settings.
- Analysis Menu, on page 252, information on creating additional custom timing reports for certain device technologies.

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 82, of the *User Guide* for details.



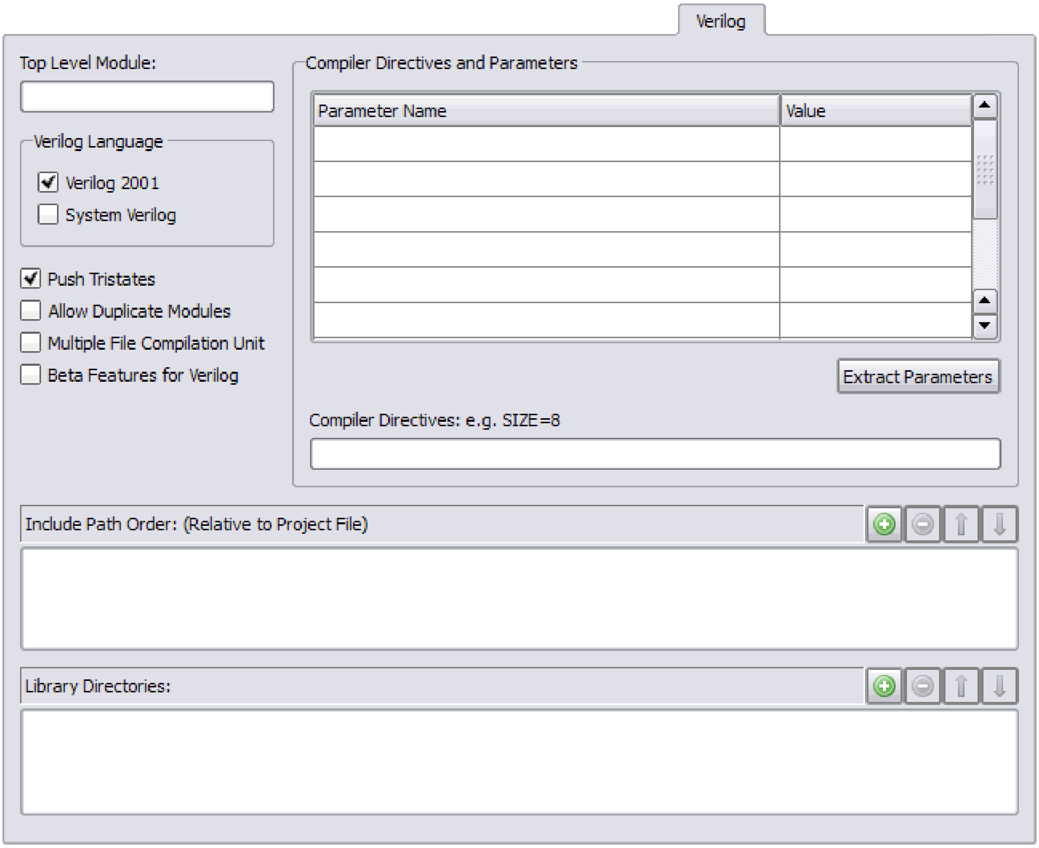
The following table describes the options available:

Feature	Description
Top Level Entity	The name of the top-level entity of your design. 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 170 for the Set VHDL Library command, or the File Options Popup Menu Command, on page 321. Tcl equivalent: **set_option -top_module** *topLevelName*
Default Enum Encoding	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 syn_encoding attribute. Tcl equivalent: **set_option -default_enum_encoding** *encodingType*
Push Tristates	When enabled (default), tristates are pushed across process/block boundaries. For more information, see Push Tristates Option, on page 202. Tcl equivalent: **set_option -compiler_compatible 0|1**
Synthesis On/Off Implemented as Translate On/Off	When enabled, the software ignores any VHDL code between synthesis_on and synthesis_off directives. It treats these third-party directives like translate_on/translate_off directives (see translate_off/translate_on, on page 180 for details). Tcl equivalent: **set_option -synthesis_onoff_pragma 0|1**
VHDL 2008	When enabled, allows you to use VHDL 2008 language standards. Tcl equivalent: **set_option -vhdl2008 0|1**
Implicit Initial Value Support	When enabled, the compiler passes init values through a syn_init property to the mapper. For more information, see VHDL Implicit Data-type Defaults, on page 485. Tcl equivalent: **set_option -supporttypedflt 0|1**

Feature	Description
Beta Features for VHDL	Enables use of any VHDL beta features included in the release. Enabling this checkbox is equivalent to including a set_option -hdl_define -set _BETA_FEATURES_ON_ directive in the project file. Tcl equivalent: **set_option -beta_vhfeatures 0|1**
Generics	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.
Extract Generic Constants	Extracts generics from the top-level entity and displays them in the table.

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 82 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** *moduleName*
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.

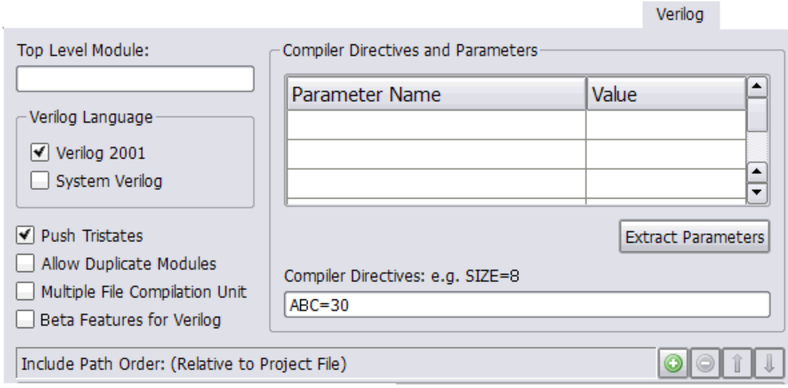
Feature	Description
Extract Parameters	Extracts design parameters from the top-level module and displays them in the table. See Compiler Directives and Design Parameters, on page 197.
Compiler Directives	Provides an interface where you can enter compiler directives that you would normally enter in the code with ‘ifdef and ‘define statements. See Compiler Directives and Design Parameters, on page 197.
Verilog Language – Verilog 2001	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 294. 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 321). Tcl equivalent: **set_option -vlog_std v2001**
Verilog Language – SystemVerilog	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. Tcl equivalent: **set_option -vlog_std sysv**
Push Tristates	When enabled (default), tristates are pushed across process/block boundaries. For details, see Push Tristates Option, on page 202. Tcl equivalent: **set_option -compiler_compatible 0|1**
Allow Duplicate Modules	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. 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. Tcl equivalent: **set_option -dup 0|1**
Multiple File Compilation Unit	When enabled (the default), the Verilog compiler uses the compilation unit for modules defined in multiple files. See SystemVerilog Compilation Units, on page 421 for additional information. Tcl equivalent: **set_option -multi_file_compilation_unit 0|1**

Feature	Description
Beta Features for Verilog	Enables use of any Verilog beta features included in the release. Enabling this checkbox is equivalent to including a set_option -hdl_define -set _BETA_FEATURES_ON_ directive in the project file. Tcl equivalent: **set_option -beta_vfeatures 0|1**
Include Path Order	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 *User Guide* for additional information.
Library Directories	Specifies all the paths to the directories which contain the Verilog library files to be included in your design for the project.

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.

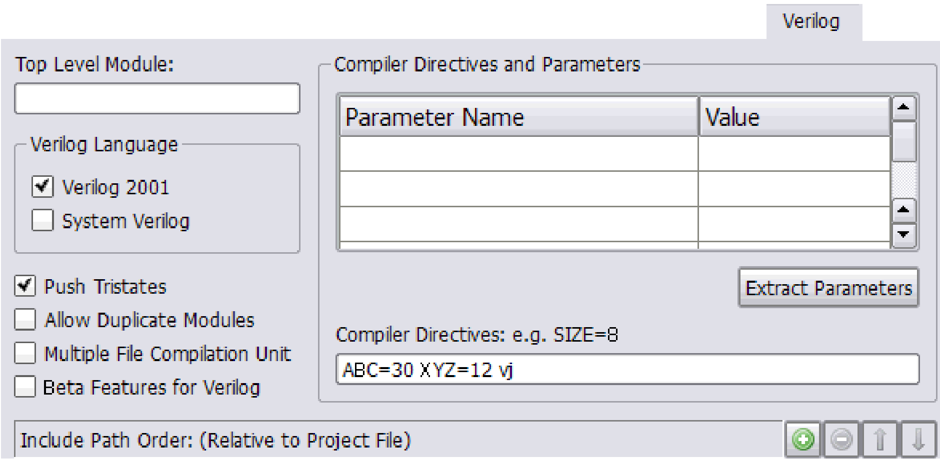
The Compiler Directives field provides an interface where you can enter compiler directives that you would normally enter in the code using ‘ifdef and ‘define statements. Use spaces to separate the statements. The directives you enter are stored in the project file. For example, if you enter the directive 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 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:

- IGNORE_VERILOG_BLACKBOX_GUTS Directive

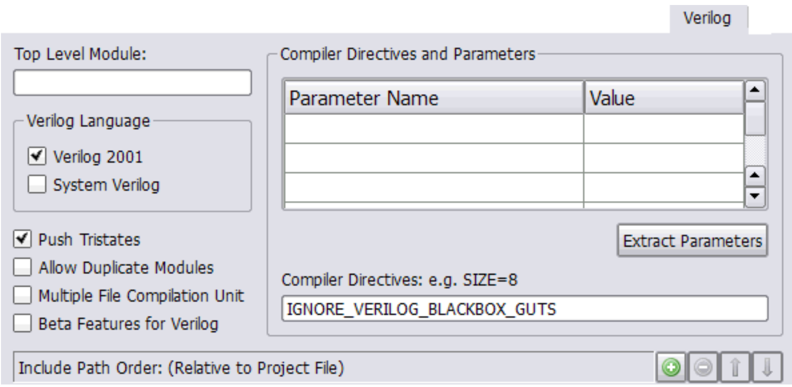
- _BETA_FEATURES_ON_ Directive
- _SEARCHFILENAMEONLY_ Directive

IGNORE_VERILOG_BLACKBOX_GUTS Directive

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.

However, if you do not want the tool to check for illegal syntax in 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:

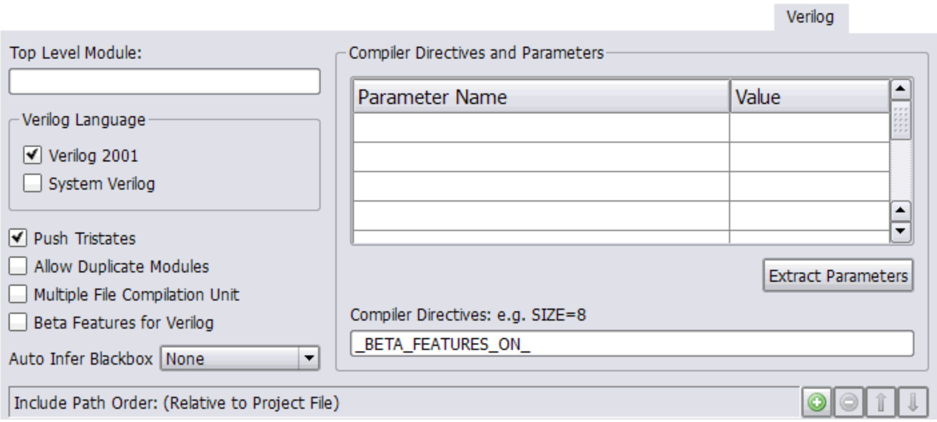
```
`define IGNORE_VERILOG_BLACKBOX_GUTS
module b1_fpga1 (A,B,C,D) /* synthesis syn_black_box */;
input B;
output A;
input [2:0] D;
output [2:0] C;
temp;
assign A = B;
assign C = D;

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_ Directive

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.



SEARCHFILENAMEONLY Directive

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"
   ```

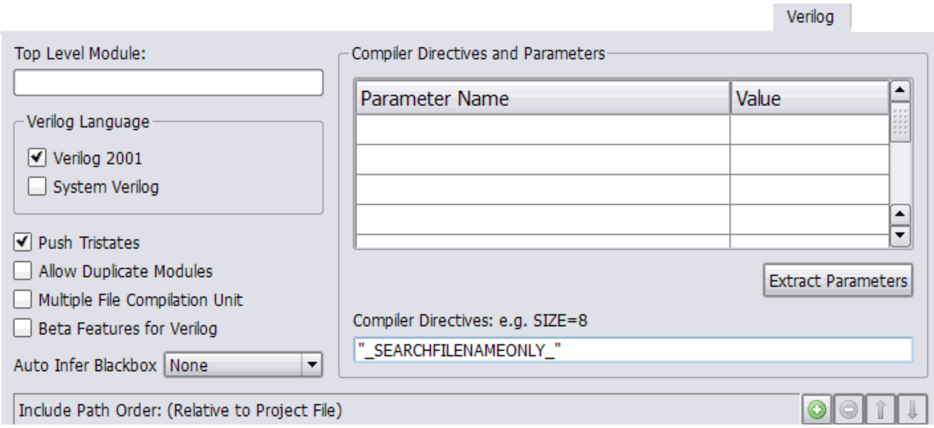
Use the _SEARCHFILENAMEONLY_ directive to resolve categories 1 and 2 above. Category 3 is a known limitation; in this case it is recommended that you adopt standard coding practices to avoid files with the same name and different content.

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:

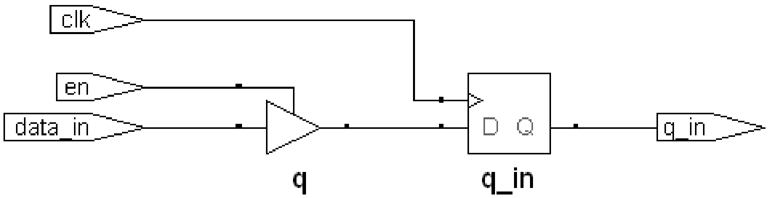
```
@W: | Macro _SEARCHFILENAMEONLY_ is set: fileName not found
attempting to search for base file name fileName
```

Push Tristates Option

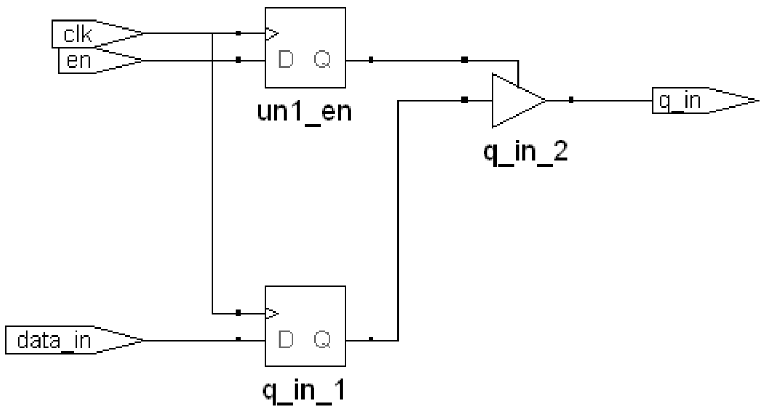
Pushing tristates is a synthesis optimization option you set with Project->Implementation Options->Verilog or VHDL.

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.



Push Tristates off:
tristate is not pushed
through the flip-flop



Push Tristates on:
tristate is pushed
through the flip-flop
so that the result
matches RTL
simulation

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.

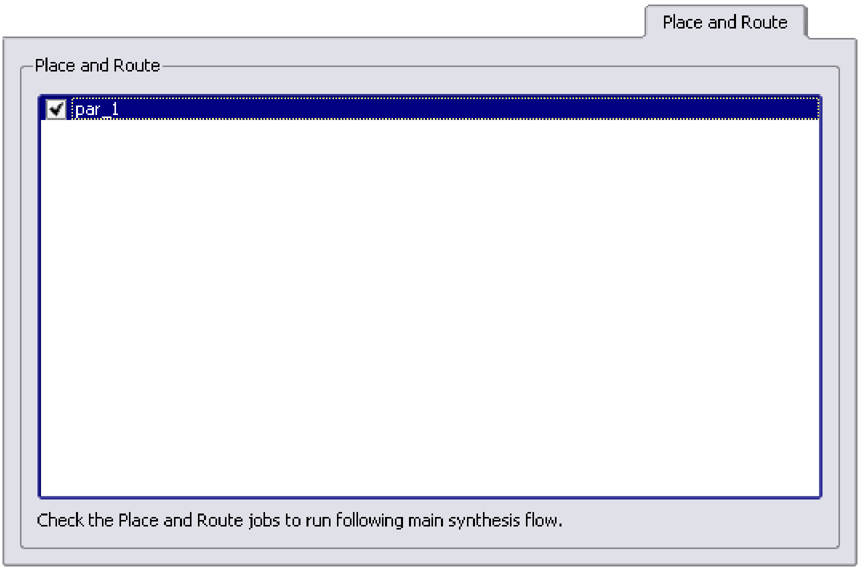
The disadvantage to pushing tristates is that you might use more design resources.

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.

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 325 or Options for Place & Route Jobs Popup Menu Command, on page 327 for details. The Place and Route Panel is only available with certain technologies.



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	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. Same as clicking the Run button in the Project view. Tcl equivalent: **project -run**
Resynthesize All	Resynthesizes (compiles and maps) the entire design, including the top level and *all compile points*, whether or not their constraints, implementation options, or source code changed since the last synthesis. If you do *not* want to force a *recompilation of all compile points*, then use Run->Run instead. Tcl equivalent: **project -run synthesis -clean**
Compile Only	Compiles the design into technology-independent high-level structures. You can view the result in the RTL view. Tcl equivalent: **project -run compile**

Command	Description
Write Output Netlist Only	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. You can also use this command in an incremental timing analysis flow. See Generating Custom Timing Reports with STA, on page 277 for details. Tcl equivalent: **project -run write_netlist**
FSM Explorer	Analyzes finite state machines contained in a design, and selects the optimum encoding style. This menu command is not available in some views. Tcl equivalent: **project -run fsm_explorer**
Syntax Check	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. Tcl equivalent: **project -run syntax_check**
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 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. Tcl equivalent: **project -run synthesis_check**
Constraint Check	Checks the syntax and applicability of the timing constraints in the .fdc file for your project and generates a report (*projectName*_cck.rpt). 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. See Constraint Checking Report, on page 272. Tcl equivalent: **project -run constraint_check**
Arrange VHDL files	Reorders the VHDL source files for synthesis. Tcl equivalent: **project -run hdl_info_gen fileorder**

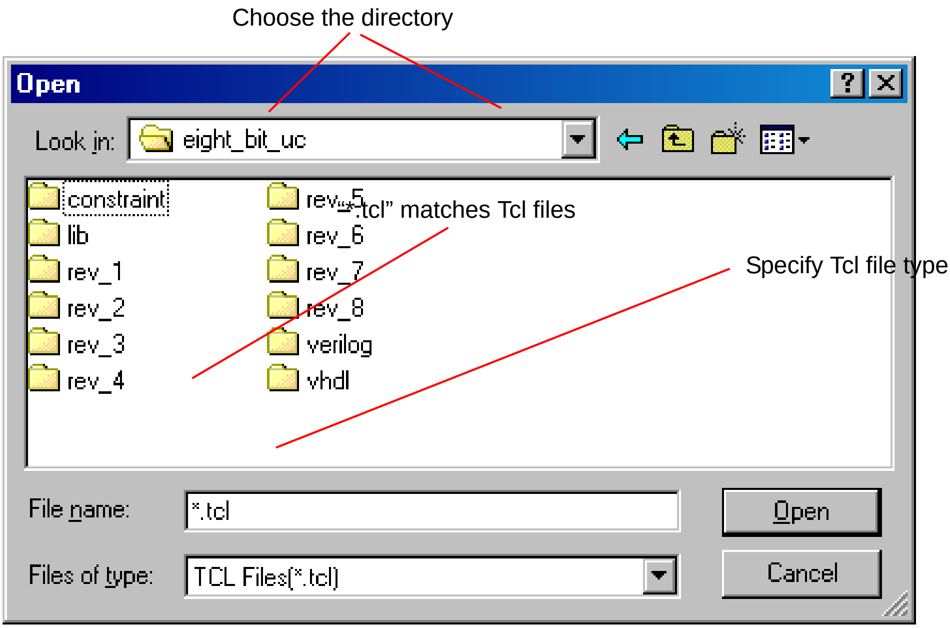
Command	Description
Launch Identify Instrumentor	Launches the Identify instrumentor. For more information, see: Working with the Identify Tools, on page 493 of the *User Guide*. To launch the Identify instrumentor in batch mode, use the set_option -identify_debug_mode 1 Tcl command.
Launch Identify Debugger	Launches the Identify debugger tool. For more information, see: Working with the Identify Tools, on page 493 of the *User Guide*. To launch the Identify debugger in batch mode, use the set_option -identify_debug_mode 1 Tcl command.
Launch SYNCore	Opens the Synopsys FPGA IP Core Wizard. This tool helps you build IP blocks such as memory or FIFO models for your design. See the Launch SYNCore Command, on page 213 for details.
Configure and Launch VCS Simulator	Allows you to configure and launch the VCS simulator. See Configure and Launch VCS Simulator Command, on page 242.
Run Tcl Script	Displays the Open dialog box, where you choose a Tcl script to run. See Run Tcl Script Command, on page 208.
Run All Implementations	Runs all selected implementations for one project at the same time. See Run All Implementations Command, on page 209. Tcl equivalent: **run -impl "***implementation1 implementation2*...**" -parallel**
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 210. 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 196.

Run Tcl Script Command

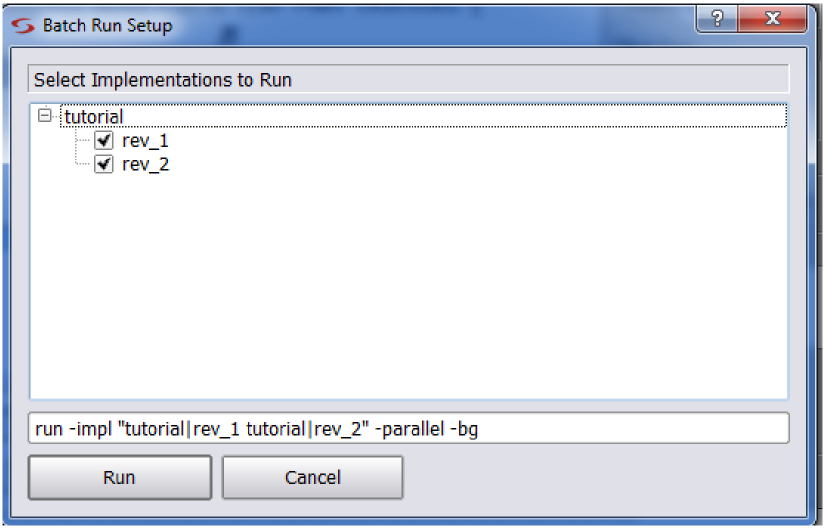
Select Run->Run Tcl Script to display the Open dialog box, where you specify the Tcl script file to execute. The File name area is filled automatically with the wildcard string “*.tcl”, corresponding to Tcl files.

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 150, for an explanation of the features in the Open dialog box.



Run All Implementations Command

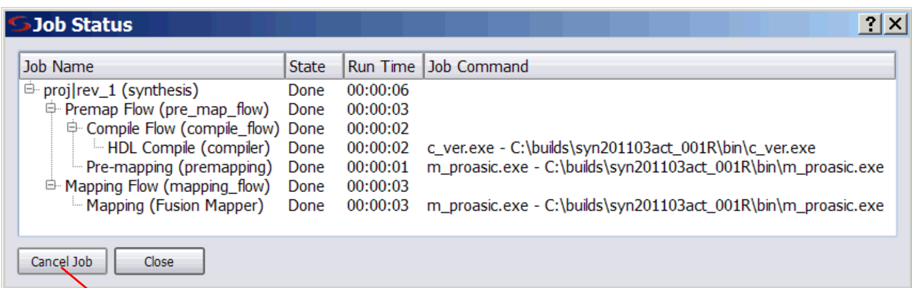
Select Run->Run All Implementations to run selected implementations in batch mode. To use the Batch Run Setup dialog box, check one or more implementations from the list displayed and click the Run button.



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 38).

You can cancel a displayed job by selecting it in the dialog box and clicking Cancel Job.



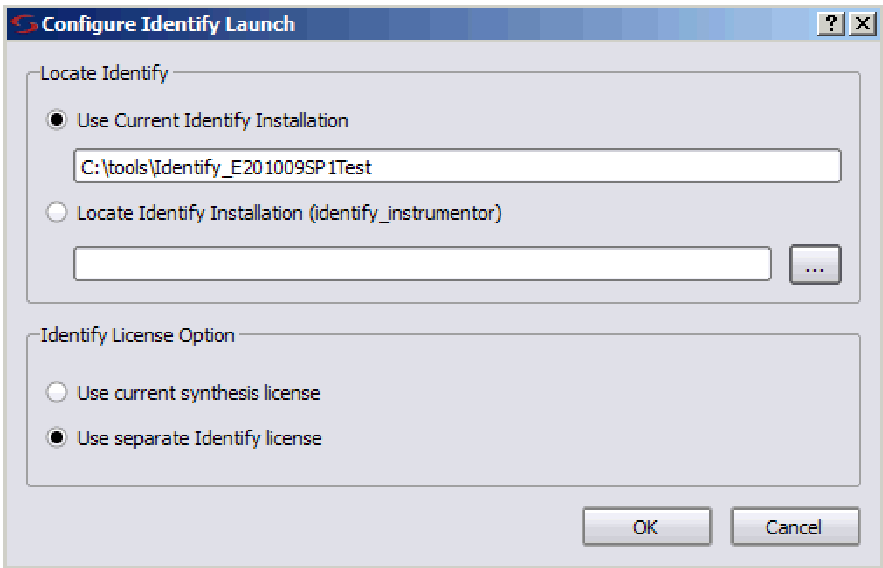
To cancel a job, select it,
then click the Cancel button

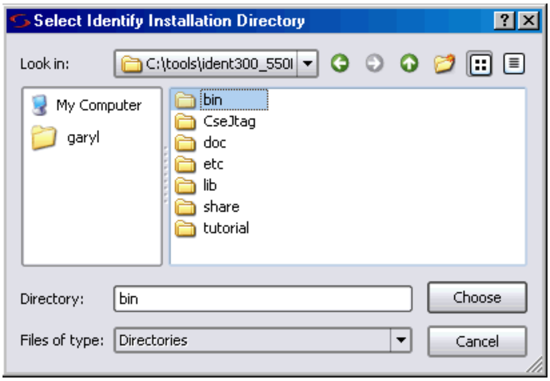
Launch Identify Instrumentor Command

The Launch Identify Instrumentor command lets you start the 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 debugger, see Working with the Identify Tools, on page 493 in the *User Guide*.

Configure Identify Launch Dialog Box

The Configure Identify Launch dialog box is automatically displayed when the location of the Identify executable has not been previously defined.



Command	Description
Use current Identify installation:	A pointer to the current installation of the Identify software. Click the radio button to use the displayed version.
Locate Identify Installation (identify_instrumentor)	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 lets you start the Identify Debugger software from within 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 493 in the *User Guide*.

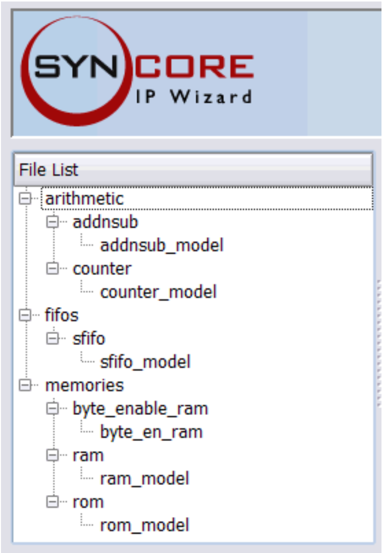
Launch SYNCore Command

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 394
- Specifying RAMs with SYNCore, on page 400
- Specifying Byte-Enable RAMs with SYNCore, on page 408
- Specifying ROMs with SYNCore, on page 414
- Specifying Adder/Subtractors with SYNCore, on page 419
- Specifying Counters with SYNCore, on page 426

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 215.
- Select ram_model and click Ok to start the RAM wizard, described in SYNCore RAM Wizard, on page 224.
- Select byte_en_ram and click Ok to start the byte-enable RAM wizard, described in SYNCore Byte-Enable RAM Wizard, on page 228.
- Select rom_model and click Ok to start the ROM wizard, described in SYNCore ROM Wizard, on page 231.
- Select addnsub_model and click Ok to start the adder/subtractor wizard, described in SYNCore Adder/Subtractor Wizard, on page 235.
- Select counter_model and click Ok to start the counter wizard, described in SYNCore Counter Wizard, on page 239.



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 215, SYNCore RAM Wizard, on page 224, SYNCore Byte-Enable RAM Wizard, on page 228, SYNCore ROM Wizard, on page 231, SYNCore Adder/Subtractor Wizard, on page 235, or SYNCore Counter Wizard, on page 239 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.

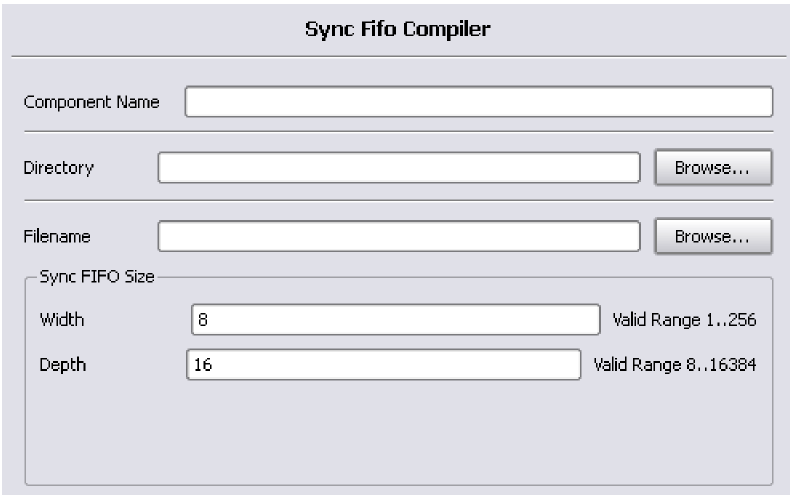
SYNCore FIFO Wizard

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 215
- SYNCore FIFO Parameters Page 2, on page 216
- SYNCore FIFO Parameters Page 3, on page 218
- SYNCore FIFO Parameters Page 4, on page 220
- SYNCore FIFO Parameters Page 5, on page 222

SYNCore FIFO Parameters Page 1

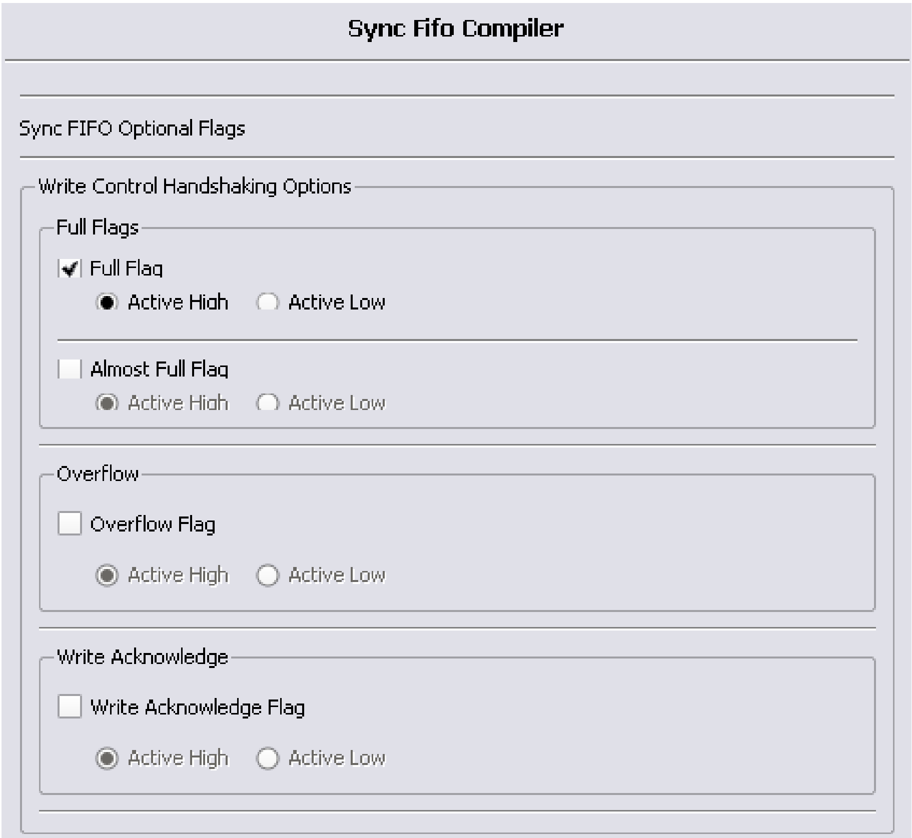
The page 1 parameters define the FIFO. Data is written/read on the rising edge of the clock.



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.

Parameter	Function
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

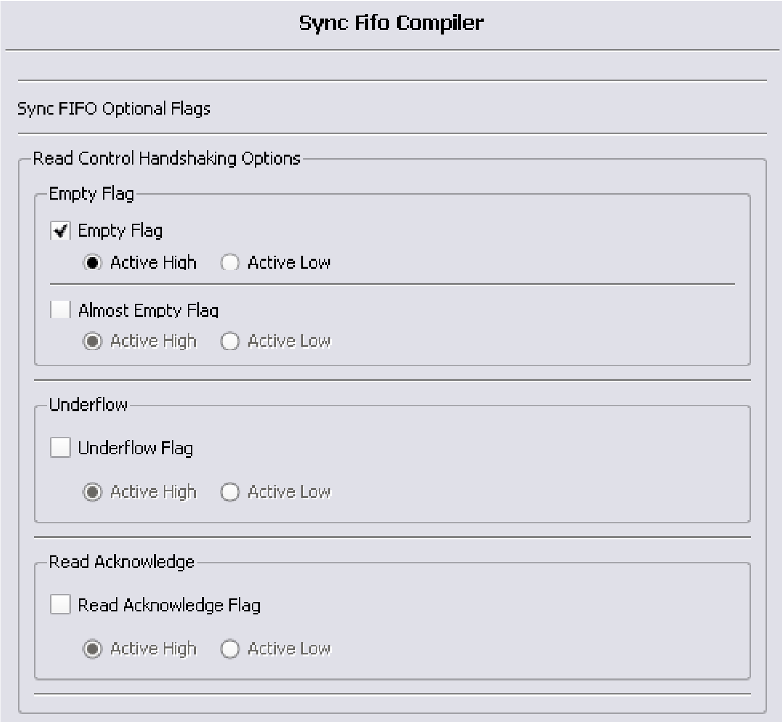


The page 2 parameters let you specify optional handshaking flags for FIFO write operations. When you specify a flag, the symbol on the left reflects your choice. Data is written/read on the rising edge of the clock.

Parameter	Function
Full Flag	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. 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 609 and FIFO Parameters, on page 607 for descriptions of the flag and parameter.
Almost Full Flag	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. 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 609 and FIFO Parameters, on page 607 for descriptions of the flag and parameter.
Overflow Flag	Specifies an Overflow signal, which is asserted to indicate that the write operation was unsuccessful because the FIFO was full. Enabling this option makes the Active High and Active Low options available for the signal (OVERFLOW_FLAG_SENSE parameter). See Handshaking Flags, on page 611 f and FIFO Parameters, on page 607 for descriptions of the flag and parameter.
Write Acknowledge Flag	Specifies a Write_ack signal, which is asserted at the completion of a successful write operation. Enabling this option makes the Active High and Active Low options (WACK_FLAG_SENSE parameter) available for the signal. See Handshaking Flags, on page 611 and FIFO Parameters, on page 607 for descriptions of the flag and parameter.
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

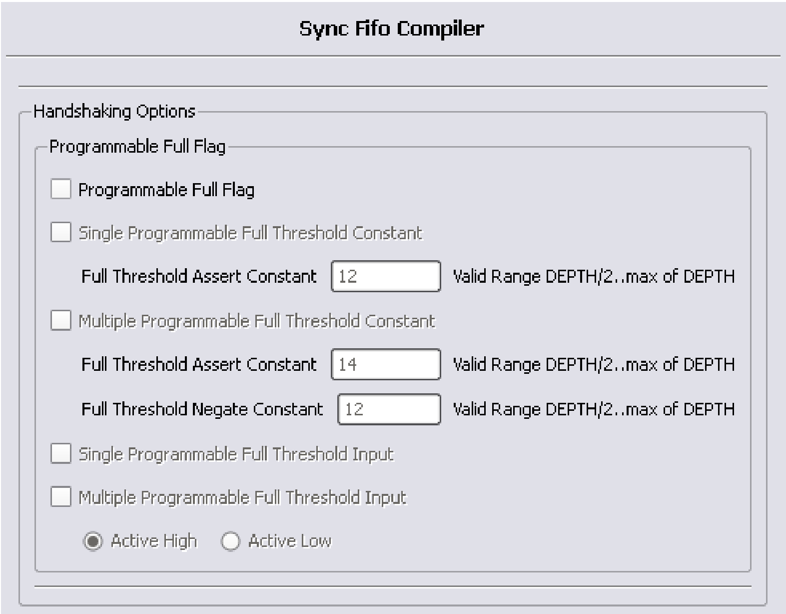
The page 3 parameters let you specify optional handshaking flags for FIFO read operations. Data is written/read on the rising edge of the clock.



Parameter	Function
Empty Flag	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. 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 610 and FIFO Parameters, on page 607 for descriptions of the flag and parameter.

Parameter	Function
Almost Empty Flag	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. 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 610 and FIFO Parameters, on page 607 for descriptions of the flag and parameter.
Underflow Flag	Specifies an Underflow signal, which is asserted to indicate that the read operation was unsuccessful because the FIFO was empty. Enabling this option makes the Active High and Active Low options (UNDRFLW_FLAG_SENSE parameter) available for the signal. See Handshaking Flags, on page 611 and FIFO Parameters, on page 607 for descriptions of the flag and parameter.
Read Acknowledge Flag	Specifies a Read_ack signal, which is asserted at the completion of a successful read operation. Enabling this option makes the Active High and Active Low options (RACK_FLAG_SENSE parameter) available for the signal. See Handshaking Flags, on page 611 and FIFO Parameters, on page 607 for descriptions of the flag and parameter.
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



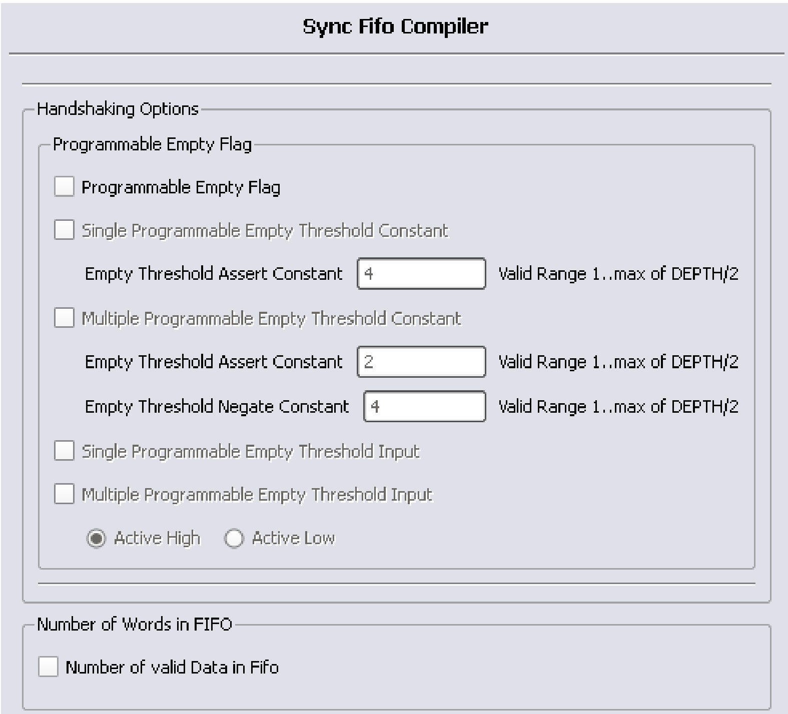
The page 4 parameters let you specify optional handshaking flags for FIFO programmable full operations. To use these options, you must have a Full signal specified. See FIFO Programmable Flags, on page 612 for details and FIFO Parameters, on page 607 for a list of the FIFO parameters. Data is written/read on the rising edge of the clock.

Parameter	Function
Programmable Full Flag	Specifies a Prog_full signal, which indicates that the FIFO has reached a user-defined full threshold. 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 613 and FIFO Parameters, on page 607 for descriptions of the flag and parameter.

Parameter	Function
Single Programmable Full Threshold Constant	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 613 for details. Enabling this option makes Full Threshold Assert Constant available.
Multiple Programmable Full Threshold Constant	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 614 for details. Enabling this option makes Full Threshold Assert Constant and Full Threshold Negate Constant available.
Full Threshold Assert Constant	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.
Full Threshold Negate Constant	Specifies a constant that is used as a threshold value for de-asserting the Prog_full signal (PGM_FULL_NTHRESH parameter).
Single Programmable Full Threshold Input	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 614 for details. Enabling this option adds the Prog_full_thresh input port to the FIFO.
Multiple Programmable Full Threshold Input	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 Constants, on page 614 for details. Enabling this option adds the Prog_full_thresh_assert and Prog_full_thresh_negate input ports to the FIFO.
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 612 for details and FIFO Parameters, on page 607 for a list of the FIFO parameters. Data is written/read on the rising edge of the clock.



Parameter	Function
Programmable Empty Flag	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 616 and FIFO Parameters, on page 607 for descriptions of the flag and parameter. 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.

Parameter	Function
Single Programmable Empty Threshold Constant	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 617 for details. Enabling this option makes Empty Threshold Assert Constant available.
Multiple Programmable Empty Threshold Constant	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 617 for details. Enabling this option makes Empty Threshold Assert Constant and Empty Threshold Negate Constant available.
Empty Threshold Assert Constant	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.
Empty Threshold Negate Constant	Specifies a constant that is used as a threshold value for de-asserting the Prog_empty signal (PGM_EMPTY_NTHRESH parameter).
Single Programmable Empty Threshold Input	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 617 for details. Enabling this option adds the Prog_full_thresh input port to the FIFO.
Multiple Programmable Empty Threshold Input	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 618 for details. Enabling this option adds the input ports to the FIFO.
Active High	Sets the specified signal to active high (1).
Active Low	Sets the specified signal to active low (0).

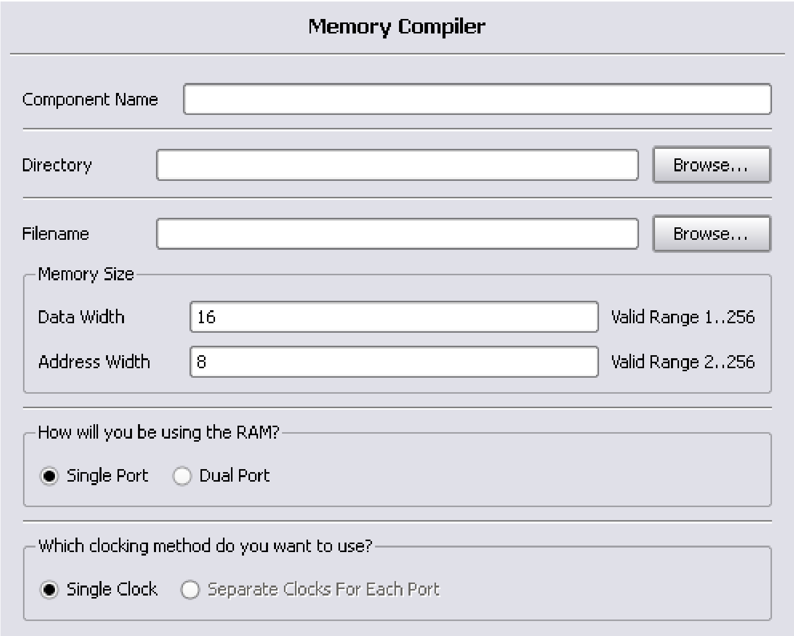
Parameter	Function
Number of Valid Data in FIFO	Specifies the Data_cnt signal for the FIFO output. This signal contains the number of words in the FIFO in the read domain.

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 224
- SYNCore RAM Parameters Pages 2 and 3, on page 226

SYNCore RAM Parameters Page 1



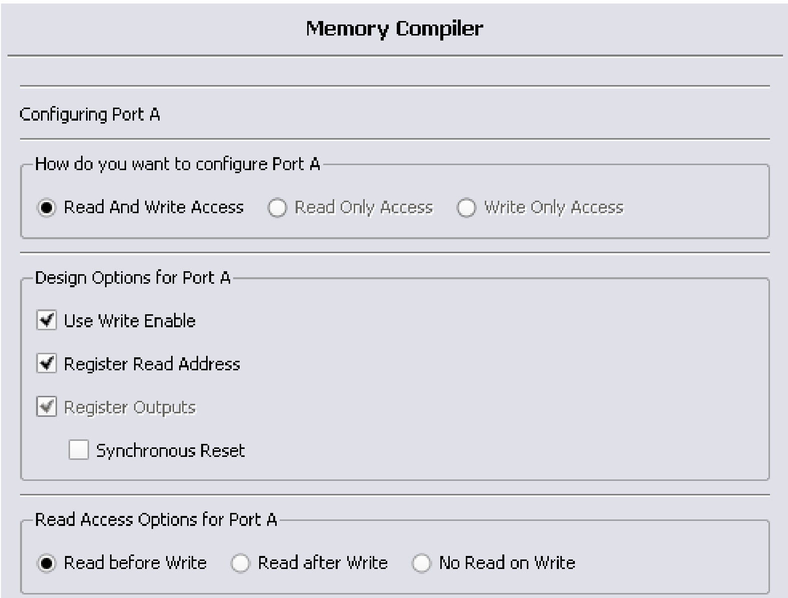
Component Name	Specifies the name of the component. This is the name that you instantiate in your design file to create an instance of the SYNCore RAM in your design. Do not use spaces. For example:
	`ram101 <ComponentName> (`
	`.PortAClk(PortAClk)`
	`, .PortAAddr(PortAAddr)`
	`, .PortADataIn(PortADataIn)`
	`, .PortAWriteEnable(PortAWriteEnable)`
	`, .PortBDataIn(PortBDataIn)`
	`, .PortBAddr(PortBAddr)`
	`, .PortBWriteEnable(PortBWriteEnable)`
	`, .PortADataOut(PortADataOut)`
	`, .PortBDataOut(PortBDataOut)`
	`);`

Directory	Specifies the directory where the generated files are stored. Do not use spaces. The following files are created: • 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_ram.v – Verilog library file required to generate RAM model • testbench.v – Verilog testbench file for testing the RAM model • instantiation_file.vin – describes how to instantiate the wrapper file • *component*.v – RAM model wrapper file generated by SYNCore Note that running the Memory Compiler wizard in the same directory overwrites the existing files.
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.



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.

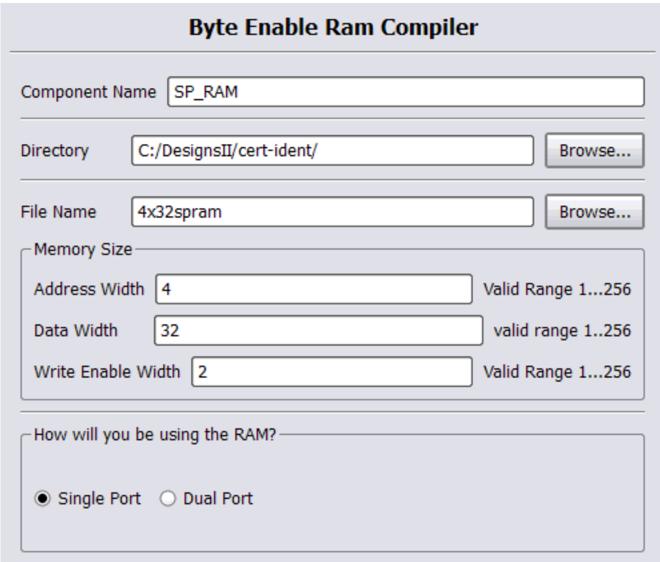
Register Read Address	Adds registers to the read address lines. The RAM symbol on the left reflects the selections you make.
Register Outputs	Adds registers to the write address lines when you specify separate read/write addressing. The register outputs are always enabled. The RAM symbol on the left 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 626.
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 627.
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 628.

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 228
- SYNCore Byte-Enable RAM Parameters Pages 2 and 3, on page 230

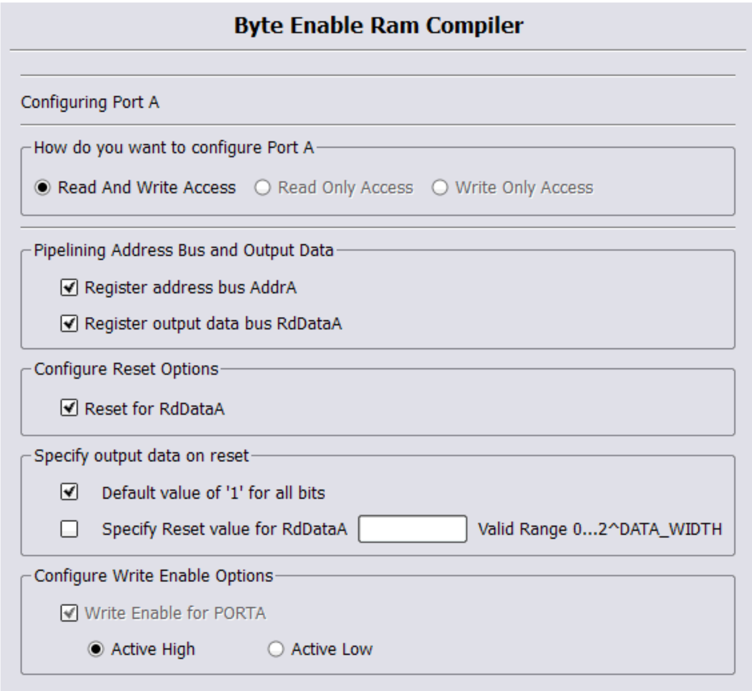
SYNCore Byte-Enable RAM Parameters Page 1



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	Specifies the directory where the generated files are stored. Do not use spaces. The following files are created: • 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 • *component*.v – Byte-enable RAM model wrapper file generated by SYNCore Note that running the byte-enable RAM wizard in the same directory overwrites the existing files.
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.



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.

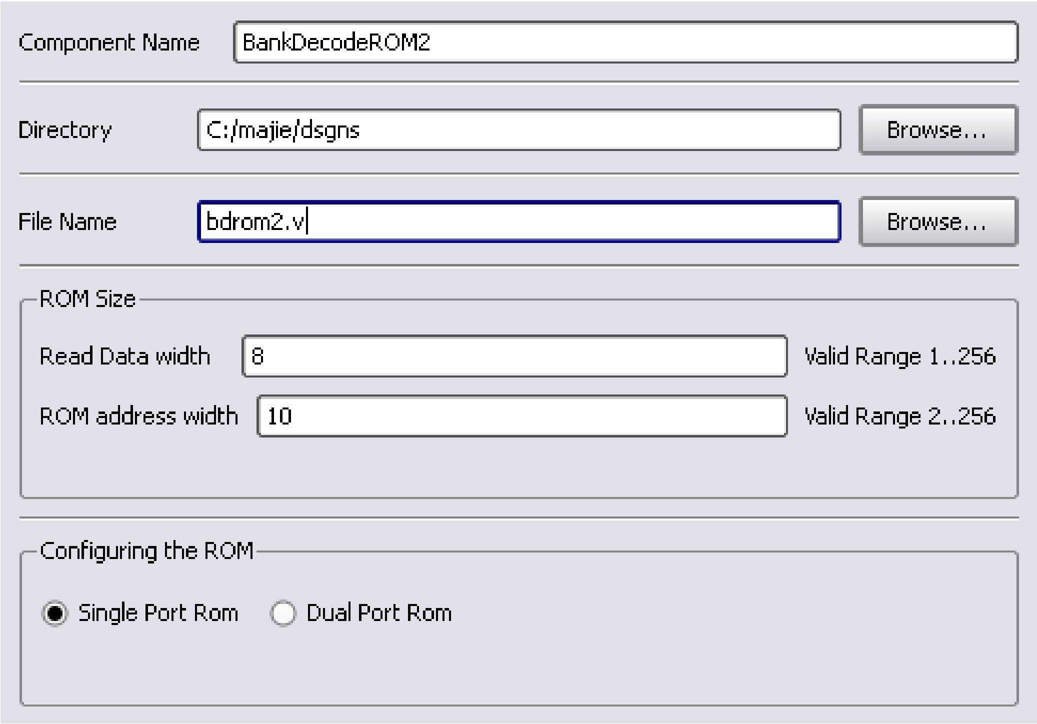
Reset for RdDataA/B	Specifies the reset type for registered read data: • Reset type is synchronous when Reset for RdDataA/B is enabled • Reset type is no reset when Reset for RdDataA/B is disabled
Specify output data on reset	Specifies reset value for registered read data (applies only when RdDataA/B is enabled): • 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	Specifies the write enable level for Port A/B. Default is Active High.

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 232
- SYNCore ROM Parameters Pages 2 and 3, on page 233
- SYNCore ROM Parameters Page 4, on page 235

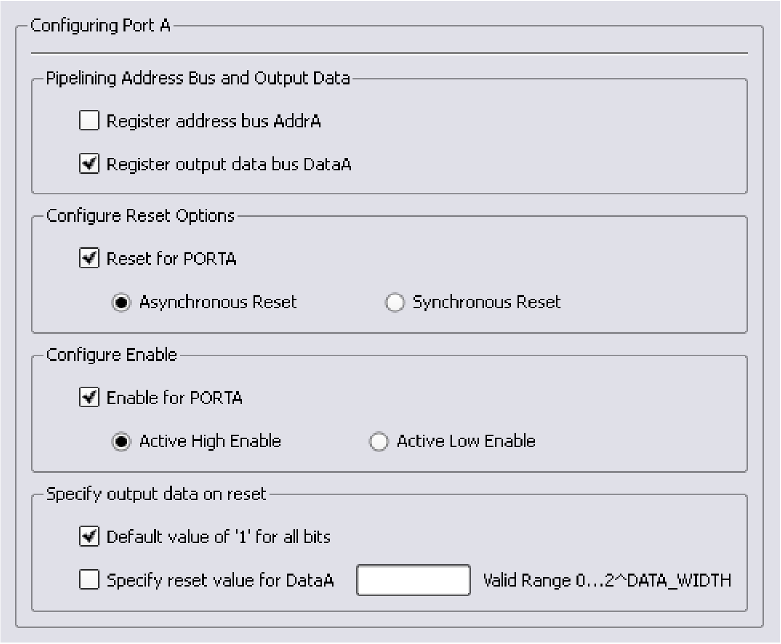
SYNCore ROM Parameters Page 1



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	Specifies the directory where the generated files are stored. Do not use spaces. The following files are created: 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_rom.v – Verilog library file required to generate ROM model testbench.v – Verilog testbench file for testing the ROM model instantiation_file.vin – describes how to instantiate the wrapper file *component*.v – ROM model wrapper file generated by SYNCore Note that running the ROM wizard in the same directory overwrites the existing files.
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

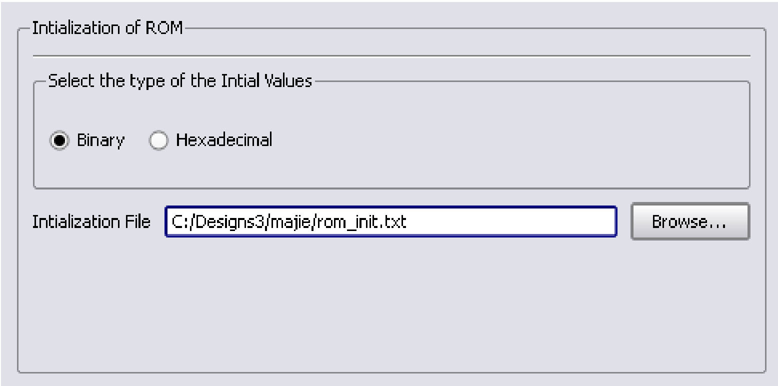
The port implementation parameters on pages 2 and 3 are the same; page 2 applies to Port A (single- and dual-port configurations), and page 3 applies to Port B (dual-port configurations only).



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



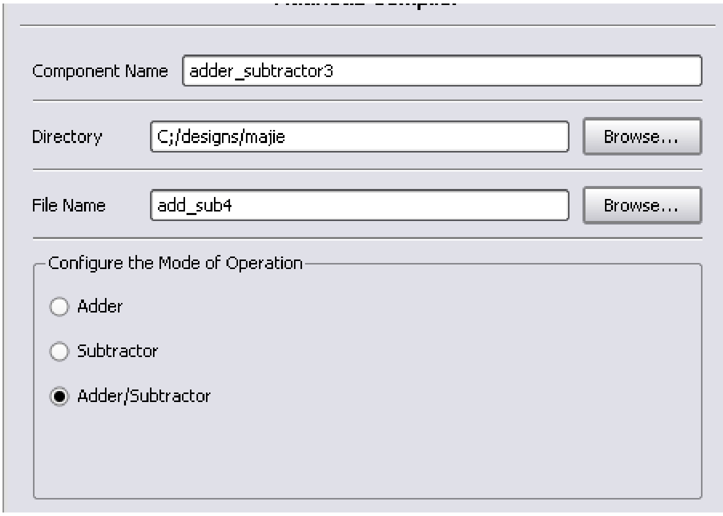
Binary	Specifies binary-formatted initialization file.
Hexadecimal	Specifies hexadecimal-formatted initial file.
Initialization File	Specifies path and filename of initialization file. The corresponding file parameter is INIT_FILE="*filename*".

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 236
- SYNCore Adder/Subtractor Parameters Page 2, on page 237

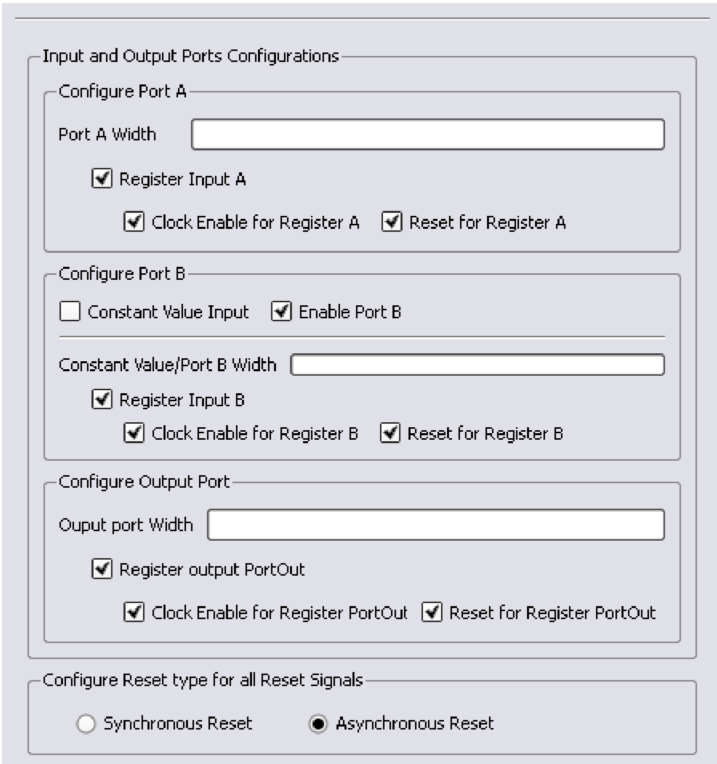
SYNCore Adder/Subtractor Parameters Page 1



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	Indicates the directory where the generated files will be stored. Do not use spaces. The following files are created: • 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 • *component*.v – adder/subtractor model wrapper file generated by SYNCore Note that running the wizard in the same directory overwrites any existing files.
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



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

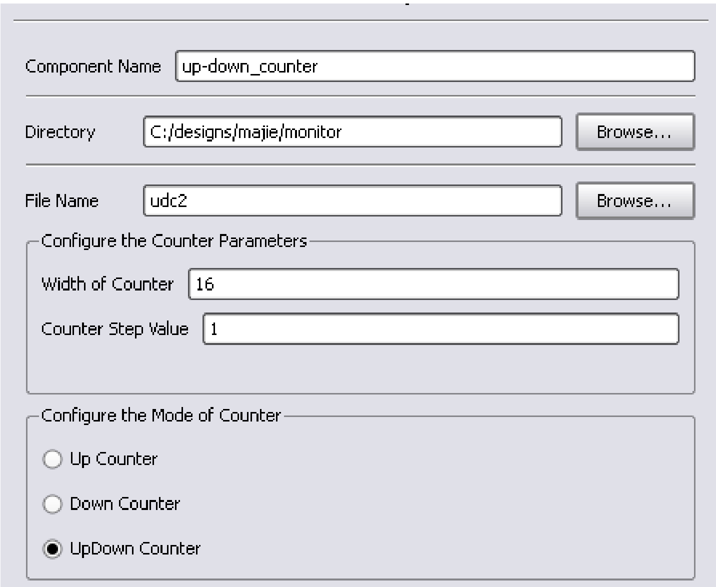
Reset for Register PortOut	Specifies the reset for the output port register
Synchronous Reset	Sets the type of reset to synchronous (the corresponding file parameter is RESET_TYPE='0')
Asynchronous Reset	Sets the type of reset to asynchronous (the corresponding file parameter is RESET_TYPE='1')

SYNCore Counter Wizard

The following describe the parameters you can set in the ROM wizard, which opens when you select counter_model:

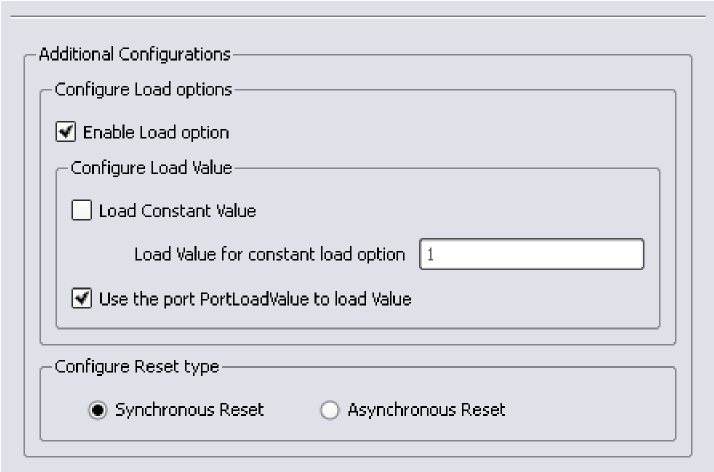
- SYNCore Counter Parameters Page 1, on page 239
- SYNCore Counter Parameters Page 2, on page 241

SYNCore Counter Parameters Page 1



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	Indicates the directory where the generated files will be stored. Do not use spaces. The following files are created: • 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_counter.v – Verilog library file required to generate counter model • testbench.v – Verilog testbench file for testing the counter model • instantiation_file.vin – describes how to instantiate the wrapper file • *component*.v – counter model wrapper file generated by SYNCore Note that running the wizard in the same directory overwrites any existing files.
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 COUNT_WIDTH=n).
Counter Step Value	Determines the counter step value (the corresponding file parameter is STEP=n).
Up Counter	Specifies an up counter (the default) configuration (the corresponding file parameter is MODE=Up).
Down Counter	Specifies an down counter configuration (the corresponding file parameter is MODE=Down).
UpDown Counter	Specifies a dynamic up/down counter configuration (the corresponding file parameter is MODE=Dynamic).

SYNCore Counter Parameters Page 2



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 tools. 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.

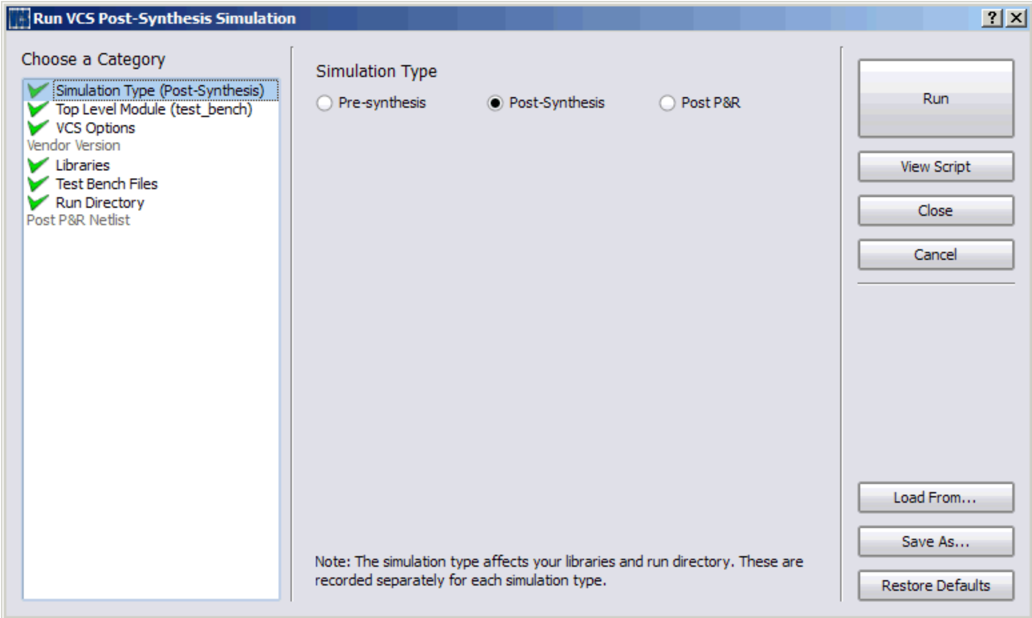
For a step-by-step procedure on setting up and launching this tool, see Simulating with the VCS Tool, on page 500 in the *User Guide*.

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 cancelling changes made in the dialog box.

Simulation Type

The following dialog box displays the Simulation Type task.


Run VCS Post-Synthesis Simulation
Choose a Category
Simulation Type (Post-Synthesis)
Top Level Module (test_bench)
VCS Options
Vendor Version
Libraries
Test Bench Files
Run Directory
Post P&R Netlist
Simulation Type
Pre-synthesis
Post-Synthesis
Post P&R
Run
View Script
Close
Cancel
Load From...
Save As...
Restore Defaults
Note: The simulation type affects your libraries and run directory. These are recorded separately for each simulation type.

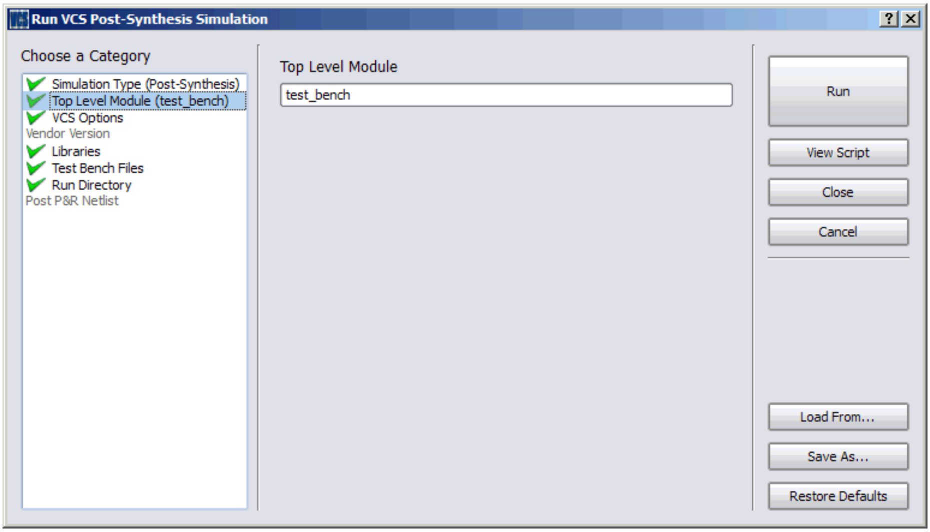
The Run VCS Simulator dialog box contains the following options:

Command	Description
Choose a Category Simulation Type	Select Simulation Type and choose the type of simulation to run: • Pre-synthesis – RTL simulation • Post-synthesis – Post-synthesis netlist simulation • Post-P&R – Post-P&R netlist simulation See Simulation Type, on page 242 to view the dialog box.
Choose a Category Top Level Module	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. See Top Level Module, on page 246 to view the dialog box.
Choose a Category VCS Options	Select VCS Options and specify options for each VCS step: • Verilog compiler – VLOGAN command options for compiling and analyzing Verilog, like 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. The default settings use the FPGA version of VCS and open the VCS GUI for the debugger (DBE) and the waveform viewer. See VCS Options, on page 246 to view the dialog box.
Choose a Category Libraries	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: • Add a library • Edit the selected library • Remove the selected library See Libraries, on page 247 and Changing Library and Test Bench Files, on page 249 for more information.

Command	Description
Choose a Category Test Bench Files	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: • Add a test bench file • Edit the selected test bench file • Remove the selected test bench file See Test Bench Files, on page 248 and Changing Library and Test Bench Files, on page 249 for more information.
Choose a Category Run Directory	Select Run Directory and specify the results directory to run the VCS simulation. See Run Directory, on page 248 to view the dialog box.
Choose a Category Post P&R Netlist	Select Post P&R Netlist and specify the post place-and-route netlist to run the VCS simulation. See Post P&R Netlist, on page 249 to view the dialog box.
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 251.
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.

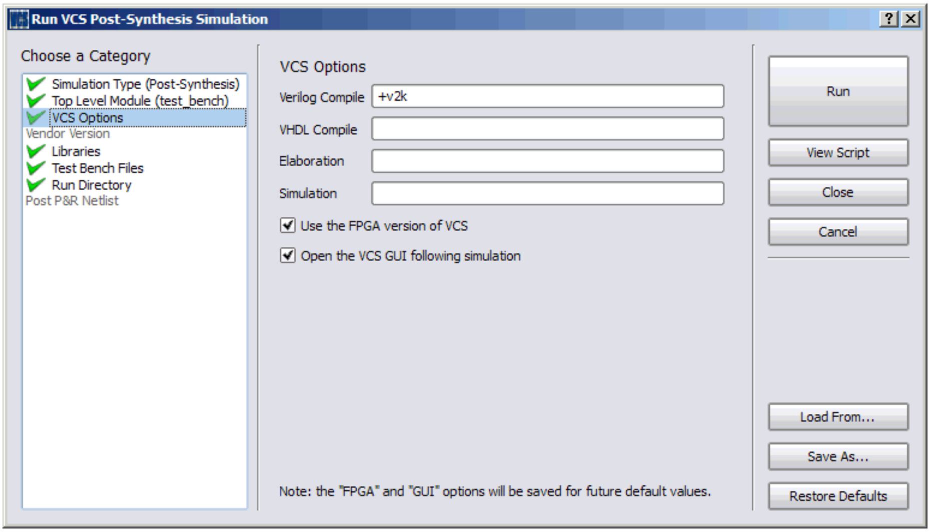
Top Level Module

The following dialog box displays the Top Level Module task.



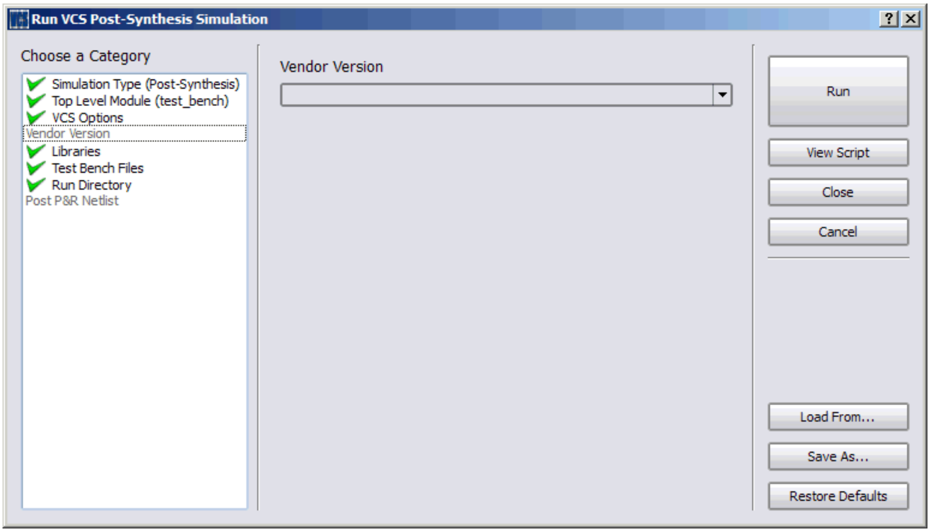
VCS Options

The following dialog box displays the VCS Options task.



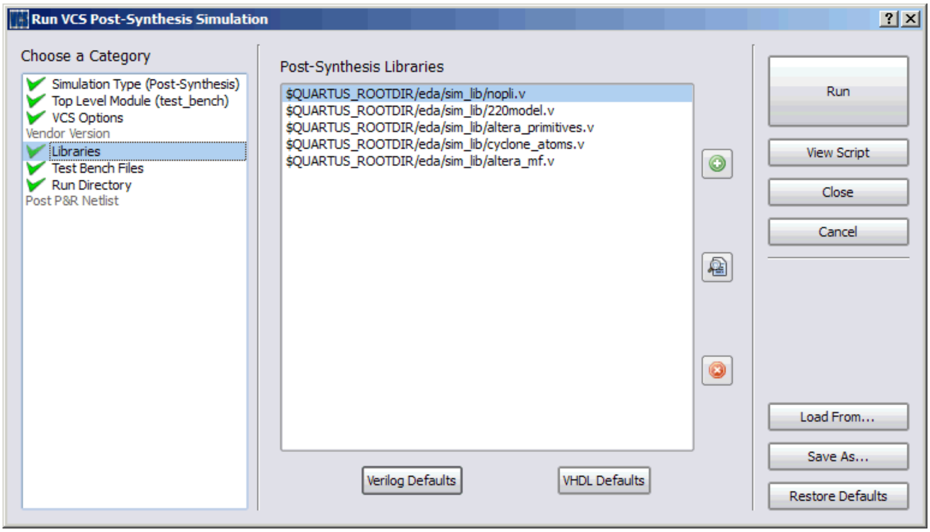
Vendor Version

The following dialog box displays the Vendor Versions task.



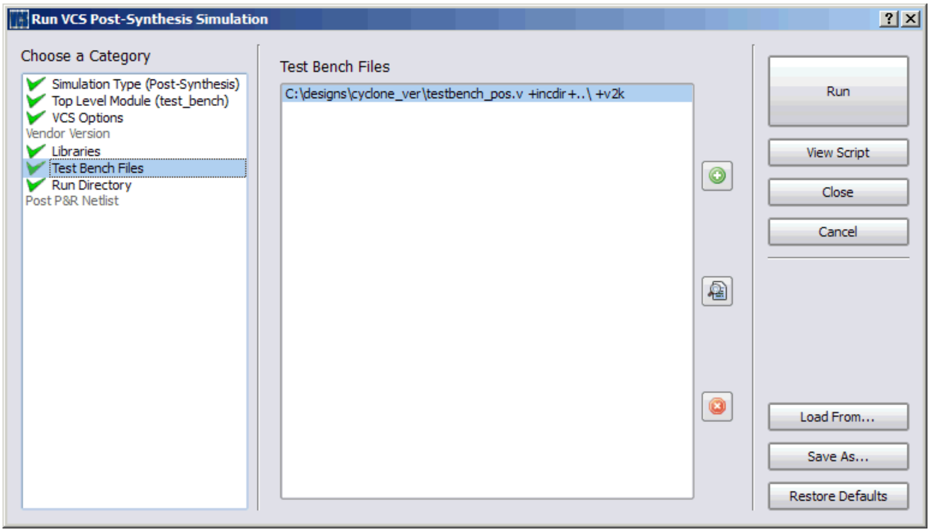
Libraries

The following dialog box displays the Libraries task.



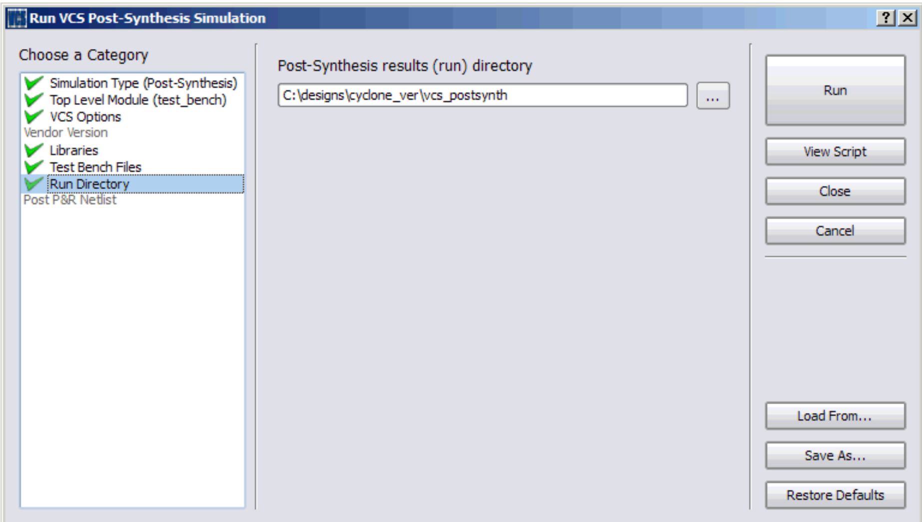
Test Bench Files

The following dialog box displays the Test Bench Files task.



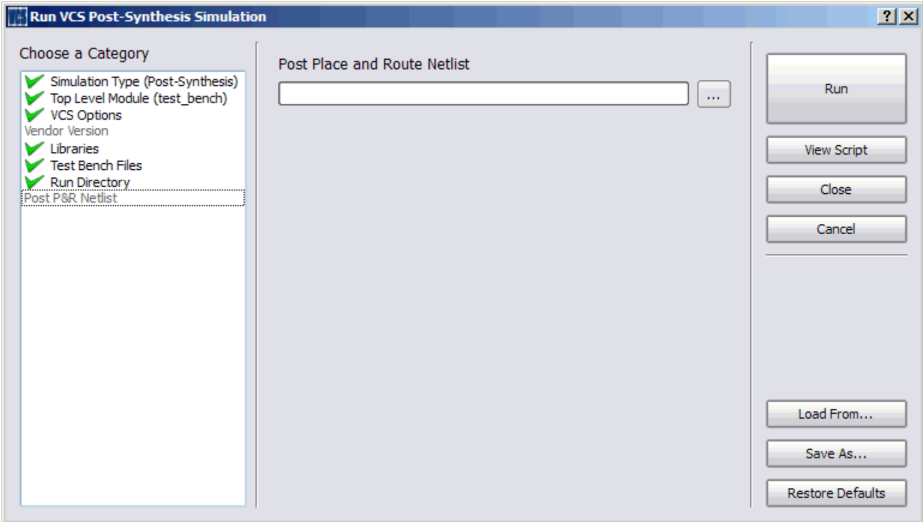
Run Directory

The following dialog box displays the Run Directory task.



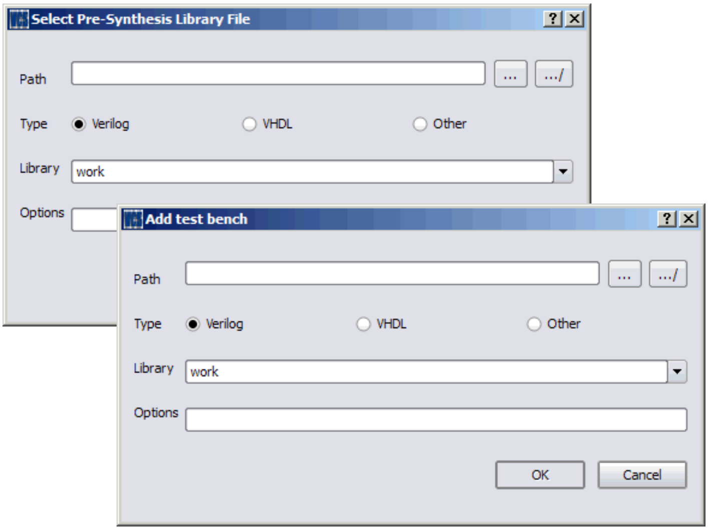
Post P&R Netlist

The following dialog box displays the Post P&R Netlist task.

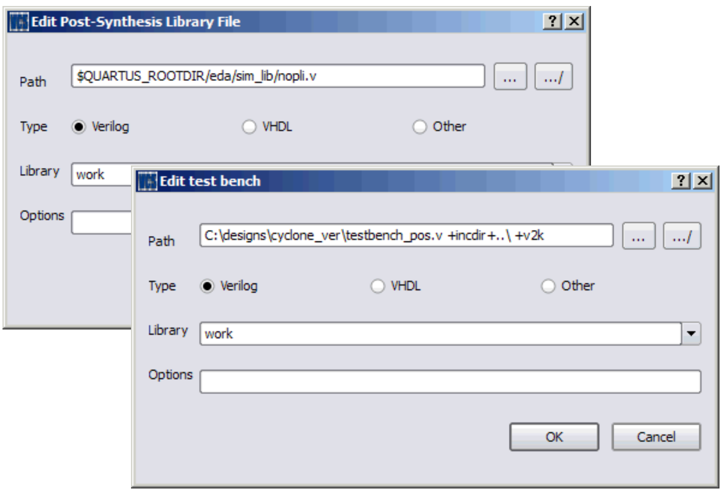


Changing Library and Test Bench Files

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 on the following dialog box.

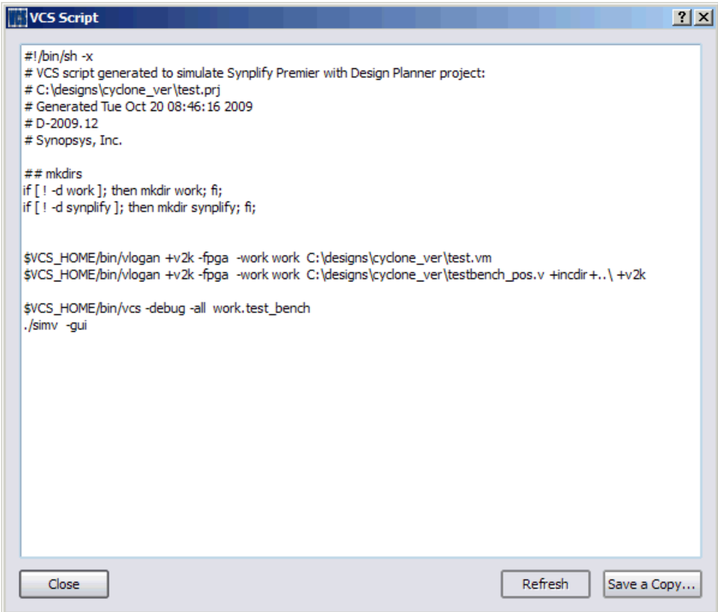


You can also edit library files and test bench files before you launch the VCS simulator. For example: specify options on the following dialog box.



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.



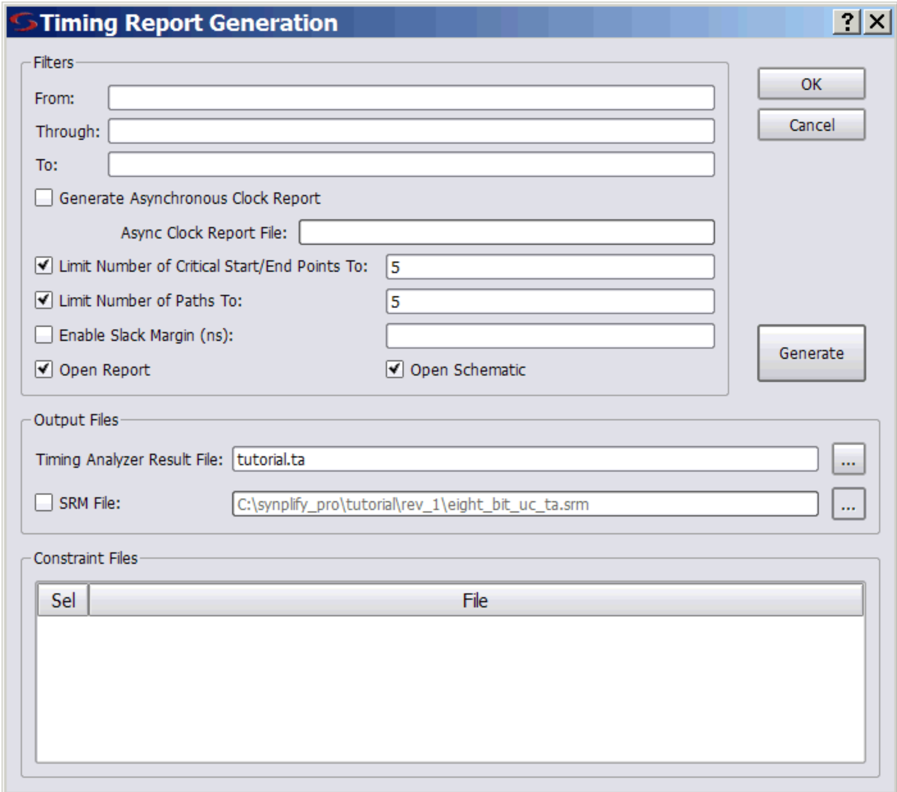
Analysis Menu

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	Displays the Timing Report Generation dialog box to specify parameters for a stand-alone customized report. See Timing Report Generation Parameters, on page 253 for information on setting these options, and Analyzing Timing in Schematic Views, on page 270 in the *User Guide* for more information. 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.
Generate Timing	Generates and displays a report using the timing option parameters specified above. See the following: • Generating Custom Timing Reports with STA, on page 277 for specifics on how to run this report. • Timing Report Generation Parameters, on page 253 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 263 for information on the file contents.



The following table provides brief descriptions of the parameters for running a stand-alone timing report.

Timing Report Option	Description
From or To	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 258 for examples of using filters. Tcl equivalent: **set_option -reporting_filter "-from {*object1*} -to {*object2*}"**
Through	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: • Timing Analyzer Through Points, on page 256 • Combining Path Filters for the Timing Analyzer, on page 258 Tcl equivalent: **set_option -reporting_filter "-from {*object1*} -to {*object2*} -through {*object3*}"**
Generate Asynchronous Clock Report	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 (*projectName*_async_clk.rpt.csv). For details on the report, see Asynchronous Clock Report, on page 270. Tcl equivalent: **set_option -reporting_async_clock 0|1**
Limit Number of Critical Start/End Points	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. Tcl equivalent: **set_option -num_startend_points** *numberOfPaths*
Limit Number of Paths to	Specifies the maximum number of paths to report. The default is 5. If you leave this field blank, all paths in the design are reported. Use this option in combination with the others in the Filters section of the dialog box. Tcl equivalent: **set_option -reporting_number_paths** *numberOfPaths*

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: • 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 *projectName*_async_clk.rpt.csv (name cannot be changed). • 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 *projectName*.ta. Tcl equivalent: **set_option -reporting_async_clock 0|1** • 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. For more details on any of these reports, see Timing Reports, on page 263.
Constraint Files	Enables analysis design constraint files (adc) to be used for stand-alone timing analysis only. See Input Files, on page 248 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:

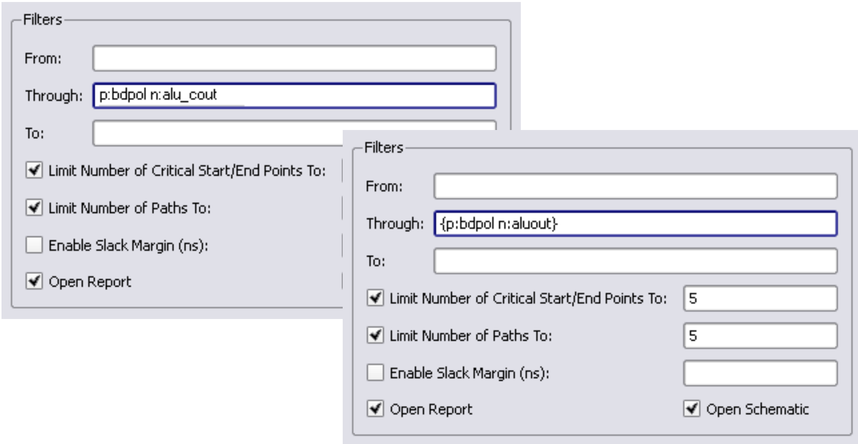
OR list	Enter the points as a space-separated list. The points are treated as an OR list and paths are reported if they crosses any of the points in the list. For example, when you type the following, the tool reports paths that pass through points b or c: `{n:b n:c}` See Filtering Points: OR List of Through Points, on page 256.

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 257.

See Defining From/To/Through Points for Timing Exceptions, on page 126 in the *User Guide* for more information about specifying through points.

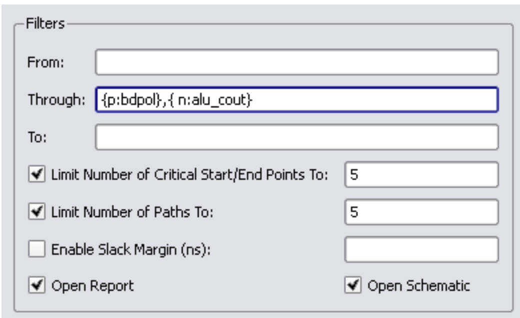
Filtering Points: OR List of Through Points

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.)



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.



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.

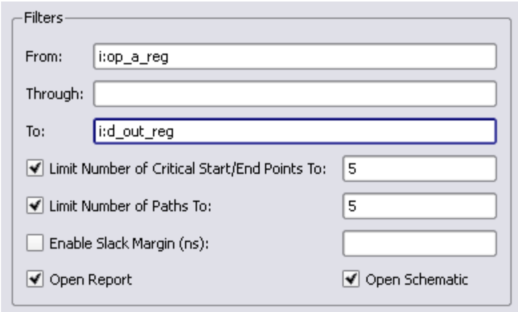
The timing analyst searches for the from/to objects in the following order: clock, port, bit port, cell (instance), net, and pin. Always use the prefix qualifier to ensure that all expected paths are reported. Remember that the timing analyst stops at the first occurrence of an object match. For buses,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

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

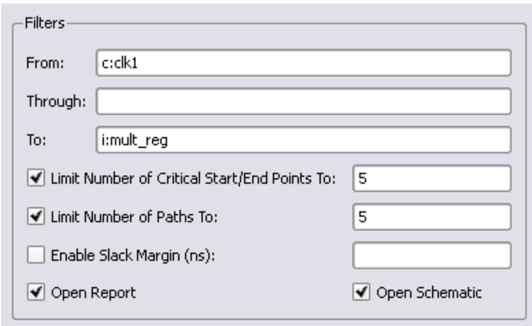
You can simply type in from/to or through points. You can also cut-and-paste or drag-and-drop valid objects from the RTL or Technology views into the appropriate fields on the Timing Report Generation dialog box. Timing analysis requires that constraints use the Tech View name space. Therefore, it is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

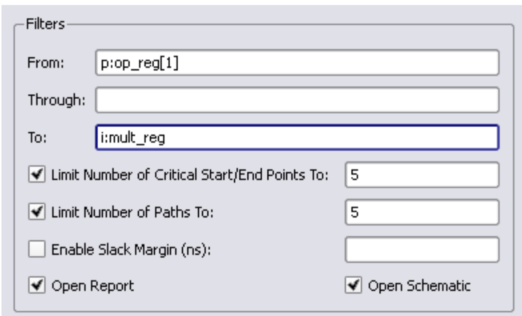
Filtering Points: Single Register to Single Register



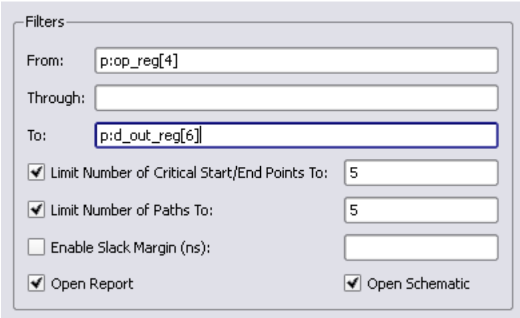
Filtering Points: Clock Object to Single Register



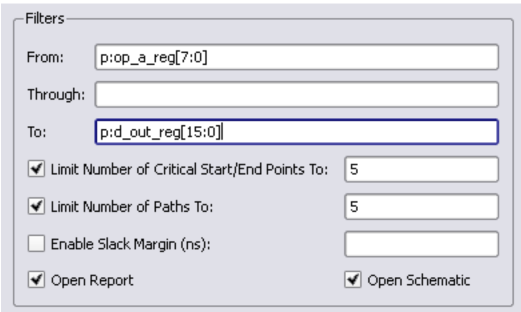
Filtering Points: Single Bit of a Bus to Single Register



Filtering Points: Single Bit of a Bus to Single Bit of a Bus

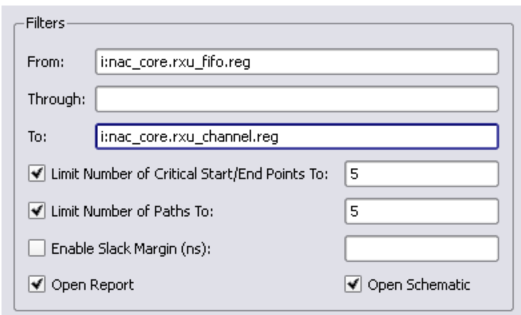


Filtering Points: Multiple Bits of a Bus to Multiple Bits of a Bus

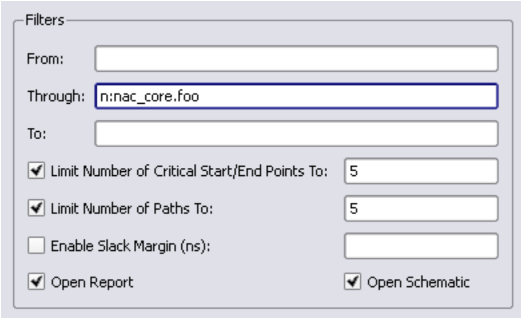


Filtering Points: With Hierarchy

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

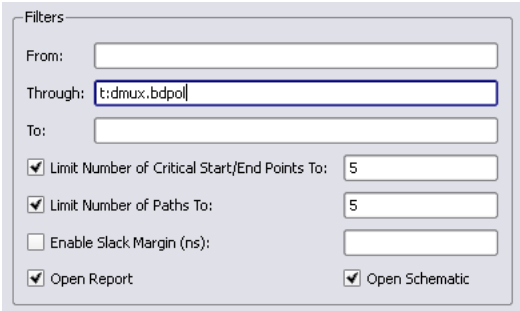


Filtering Points: Through Point for a Net



Filtering Points: Through Point for a Hierarchical Port

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



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

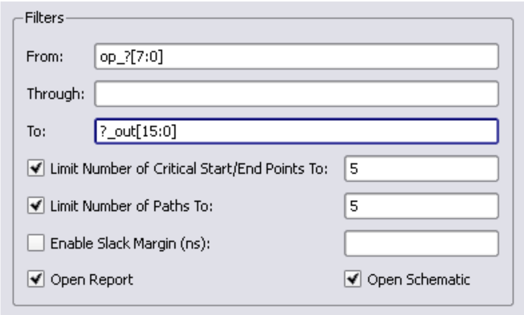
The ? matches single characters. If a design has buses op_a[7:0], op_b[7:0], and op_c[7:0], and you want to filter the paths starting at each of these buses, specify the start points as op_?[7:0]. See Example: ? Wildcard in the Name, on page 262 for another example.

The * Wildcard

The * matches a string of characters. In a design with buses op_a2[7:0], op_b2[7:0], and op_c2[7:0], where you want to filter the paths starting at each of these objects, specify the start points as op_*[*]. The report shows all paths beginning at each of these buses and for all of the bits of each bus. See Example: * Wildcard in the Name (With Hierarchy), on page 263 and Example: * Wildcard in the Bus Index, on page 263 for more examples.

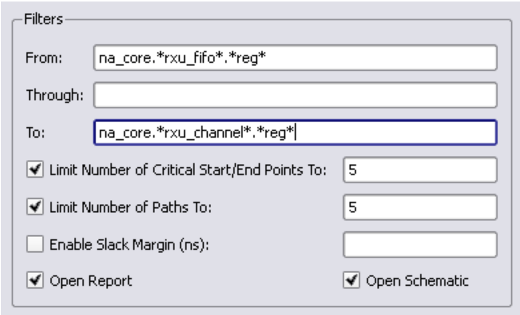
Example: ? Wildcard in the Name

The ? is not supported in bus indices.



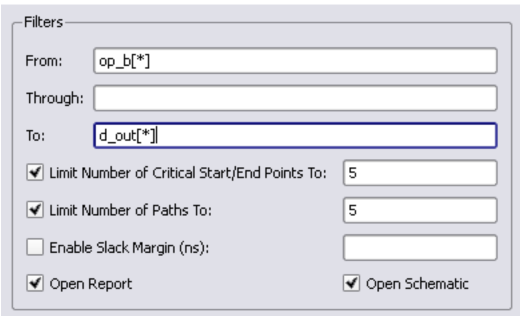
Example: * Wildcard in the Name (With Hierarchy)

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.



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.



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 106 and Transparent and Opaque Display of Hierarchical Instances, on page 111. For procedures on using the HDL Analyst tool, see Analyzing With the HDL Analyst Tool, on page 247 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 264
- HDL Analyst Menu: Hierarchical and Current Level Submenus, on page 265
- HDL Analyst Menu: Filtering and Flattening Commands, on page 267
- HDL Analyst Menu: Timing Commands, on page 271
- HDL Analyst Menu: Analysis Commands, on page 271
- HDL Analyst Menu: Selection Commands, on page 275
- HDL Analyst Menu: FSM Commands, on page 275

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 247 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 269 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 269 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 269 for tips about flattening
Technology->Hierarchical Critical Path	Creates a new Technology view of your design, with a hierarchical, *filtered* 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, *filtered* 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 269 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 247 of the *User Guide*.

HDL Analyst Command	Description
Hierarchical->Expand	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 *filtered* schematic. Operates hierarchically, on lower schematic levels as well as the current level. Successive Expand commands expand the paths further, based on the new current selection.
Hierarchical->Expand to Register/Port	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 *filtered* schematic. Operates hierarchically, on lower schematic levels as well as the current level.
Hierarchical->Expand Paths	Shows all logic, on any hierarchical level, between two or more selected instances, pins, or ports. The result is a *filtered* schematic. Operates hierarchically, on lower schematic levels as well as the current level.
Hierarchical->Expand Inwards	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 *filtered* schematic. Operates hierarchically, on lower schematic levels as well as the current level.
Hierarchical->Goto Net Driver	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.
Hierarchical->Select Net Driver	Selects the driver for the selected net. The result is a *filtered* schematic. Operates hierarchically, on lower schematic levels as well as the current level.
Hierarchical->Select Net Instances	Selects instances connected to the selected net. The result is a *filtered* schematic. Operates hierarchically, on lower schematic levels as well as the current level.
Current Level->Expand	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 *filtered* schematic. Limited to all sheets on the current schematic level. This command is only available if a HDL Analyst view is open. Successive Expand commands expand the paths further, based on the new current selection.

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 *filtered* 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 *filtered* 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 *filtered* 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 *filtered* 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 247 of the *User Guide*.

HDL Analyst Command	Description
Filter Schematic	Filters your entire design to show only the selected objects. The result is a *filtered* schematic. For more information about using this command, see Filtering Schematics, on page 251 of the *User Guide*. 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 269 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 269 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 259 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 247 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 270 on the *User Guide*.

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 247 of the *User Guide*.

HDL Analyst Command	Description
Isolate Paths	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. If the selected objects include ports and/or pins on unselected instances, the result also includes paths associated with those selected objects. 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. 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.
Show Context	Shows the original, unfiltered schematic sheet that contains the selected instance. Available only in a filtered schematic.
Hide Instances	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. 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 274 for tips.
Unhide Instances	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.

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 (HDL Analyst Options Command, on page 289). For usage tips, see Usage Notes for Dissolving Instances, on page 274.
Dissolve to Gates	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. The number of levels dissolved is determined by the Dissolve Levels value in the HDL Analyst Options dialog box (HDL Analyst Options Command, on page 289). 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. The effect of the command varies: • In an unfiltered schematic, this command *flattens* 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 *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. 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 259 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 155) 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 259 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.

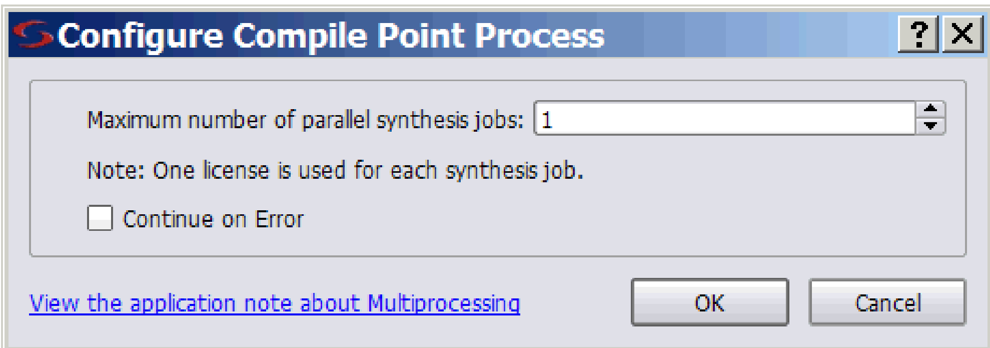
Command	Description
Basic Options Menu Commands for all Views	
Configure VHDL Compiler	Opens the Implementation Options dialog box where you can set the top-level entity and the encoding method for enumerated types. State-machine encoding is automatically determined by the FSM compiler or FSM explorer, or you can specify it explicitly using the syn_encoding attribute. See Implementation Options Command, on page 183 for details.
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 183.
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 277.
Toolbars	Lets you customize your toolbars.
Project View Options	Sets options for organizing files in the Project view. See Project View Options Command, on page 280.
Editor Options	Sets your Text Editor syntax coloring, font, and tabs. See Editor Options Command, on page 285.
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 288.
HDL Analyst Options	Sets display preferences for HDL Analyst schematics (RTL and Technology views). See HDL Analyst Options Command, on page 289.

Command	Description
Configure External Programs	Lets you set browser and Acrobat Reader options on Linux platforms. See Configure External Programs Command, on page 294 for details.
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 495 in the *User Guide*.

Configure Compile Point Process Command

Use the Configure Compile Point Process command to let you 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 grayed out for technologies that are not supported.



Field/Option	Description
Maximum Number of Parallel Synthesis Jobs	Sets the maximum number of synthesis jobs that can run in parallel. It displays the current value from the `ini` file, and allows you to reset it. Use this option for multiprocessing by running compile point jobs in parallel. Set a value based on the number of available licenses. Note that one license is used for each job. See License Utilization for Multiprocessing, on page 279. When you set this option, it resets the MaxParallelJobs value in the `.ini` file. See Maximum Parallel Jobs, on page 279 for other ways to specify this value.
Continue on Error	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. The Continue on Error mode automatically enables the MultiProcessing 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, *Using Multiprocessing* for details.

Maximum Parallel Jobs

There are three ways to specify the maximum number of parallel jobs:

`ini` File	Set this variable in the MaxParallelJobs variable in the product `ini` file:
	`[JobSetting]`
	`MaxParallelJobs=<n>`
	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.

Tcl Variable	Set the following variable in a Tcl file, the project files, or from the Tcl window: `set_option -max_parallel_jobs=<n>` 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.
Configure Compile Point Process Command	The Maximum Number of Parallel Synthesis Jobs option displays the current ini file value and allows you to reset it.

License Utilization for Multiprocessing

When you decide to run parallel synthesis jobs, a license is used for each compile point job that runs. For example, if you set the Maximum number of parallel synthesis jobs to 4, then the synthesis tool consumes one license and three additional licenses are utilized to run the 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:

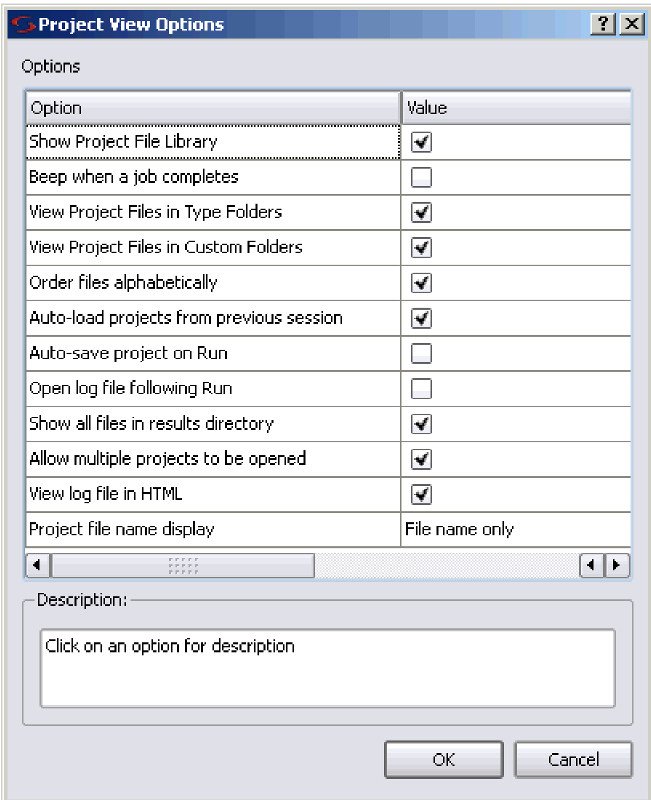
1. Synthesis software scheme for the compile point requirements used to determine the maximum number of parallel jobs or licenses a particular design tries to use.
2. Value set on the Configure Compile Point Process dialog box.

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 Specifying License Types, on page 478.

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.

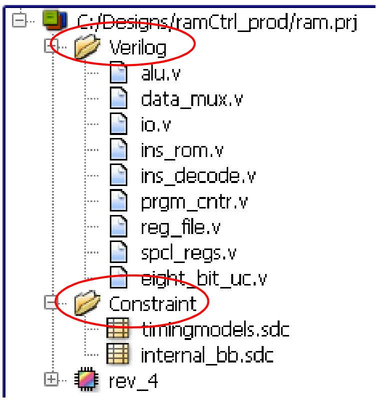


The following table describes the Project View Options dialog box features.

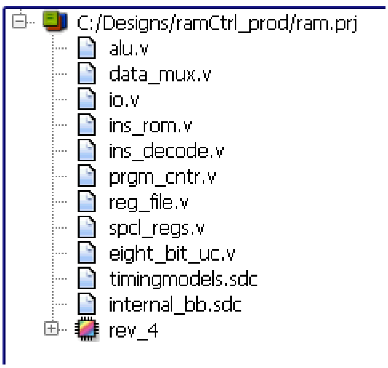
Field/Option	Description
Show Project File Library	When enabled, displays the corresponding VHDL library next to each source VHDL filename, in the Project Tree view of the Project view. For example, with library dune, file pc.vhd is listed as [dune] pc.vhd if this option is enabled, and as pc.vhd if it is disabled. See also Set VHDL Library Command, on page 174, 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 282 and add_file, on page 16.
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 283.
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 283.
Auto-save project on Run	Enable/Disable automatically saving projects when the Run button is selected. See Automatically Save Project on Run, on page 284.
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 283.
View log file in HTML	Enable/Disable viewing of log file report in HTML format versus text format. See Log File, on page 257.
Project file name display	From the drop-down menu, select one the following ways to display project files: • 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: • off—appends individual job logs to the `srr` log file. • on—always link to individual job logs. • if_up_to_date—only link to individual job logs if the module is up-to-date.

View Project Files in Type Folders Option



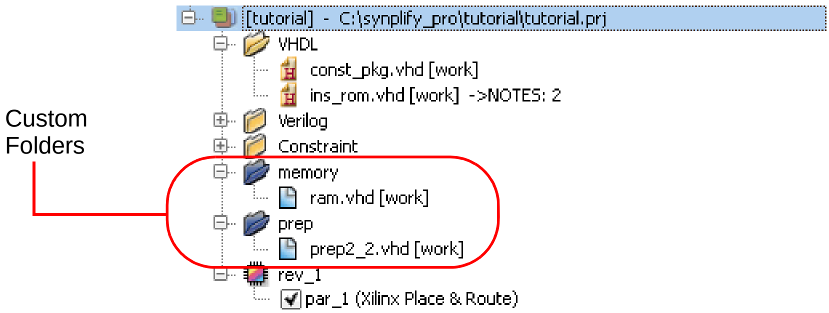
View project files in type folders *enabled*



View project files in type folders *disabled*

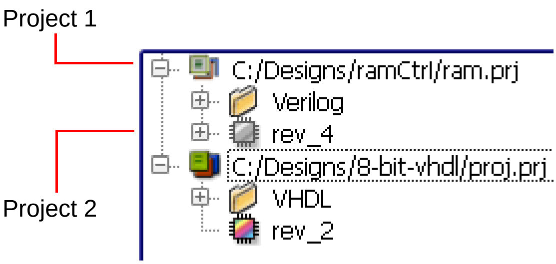
View Project Files in Custom Folders Option

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

The following figure shows multiple projects open.



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:

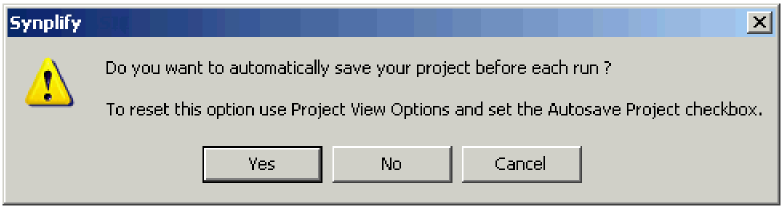
```
Project run command encountered during project load. Are you sure
you want to run?
```

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.

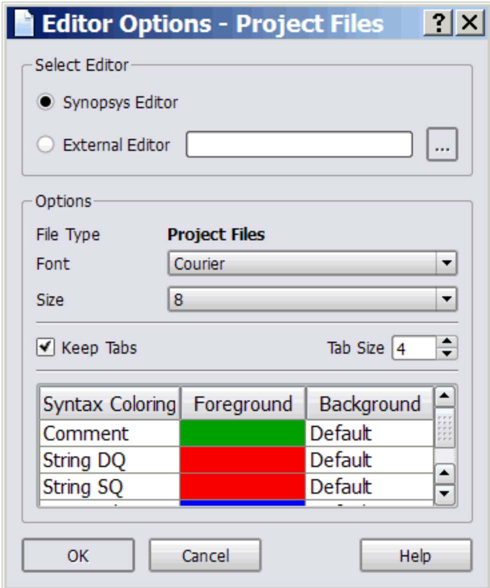


You can specify one of the following:

- 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.



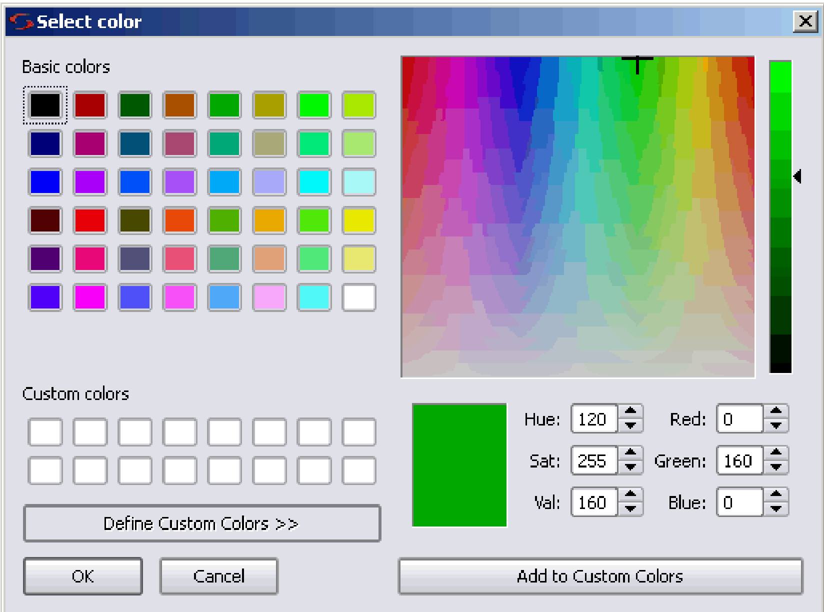
The following table describes the Editor Options dialog box features.

Feature	Description
Select Editor	Select an internal or external editor.
• Synopsys Editor	Sets the Synopsys text editor as the default text editor.
• External Editor	Uses the specified external text editor program to view text files from within the Synopsys FPGA tool. The executable specified must open its own window for text editing. See Using an External Text Editor, on page 42 of the *User Guide* for a procedure. *Note:* Files opened with an external editor *cannot* be crossprobed.
Options	Set text editing preferences.
• 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 286.

Color Options

Click in the Foreground or Background field for the corresponding object in the Syntax Coloring field to display the color palette.

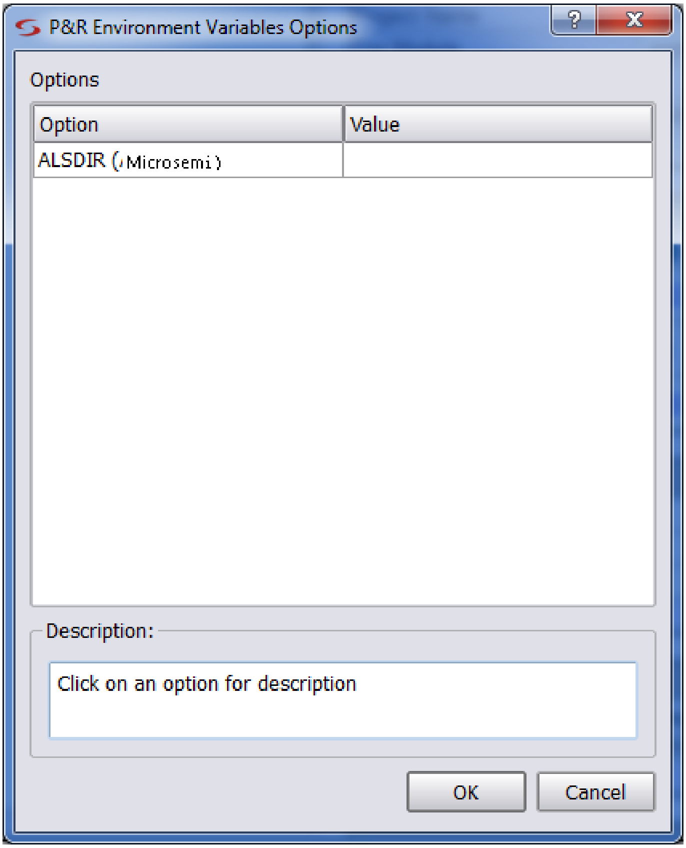


You can set syntax colors for some common syntax options listed in the following table.

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.



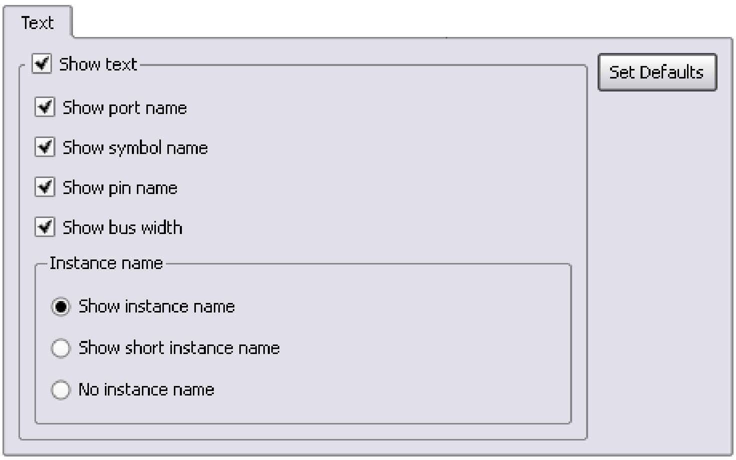
HDL Analyst Options Command

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 View Preferences, on page 215 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 289
- General Panel, on page 290
- Sheet Size Panel, on page 292
- Visual Properties Panel, on page 294

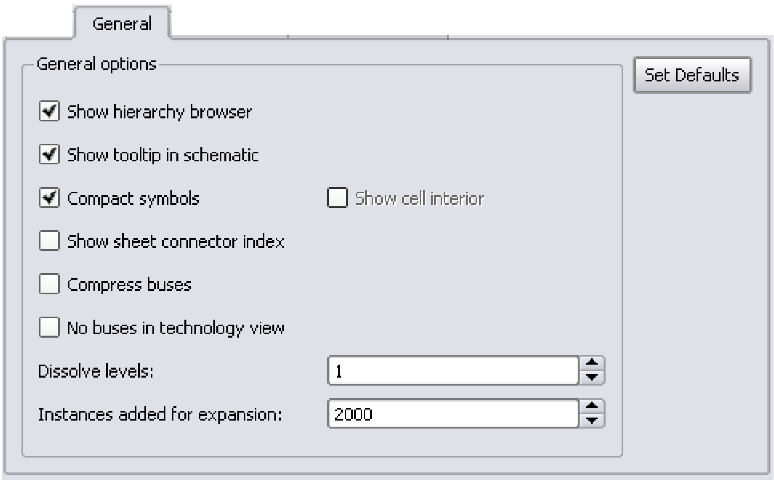
Text Panel



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: • Show instance name • Show short instance name • No instance name
Set Defaults	Set the dialog box to display the default values.

General Panel

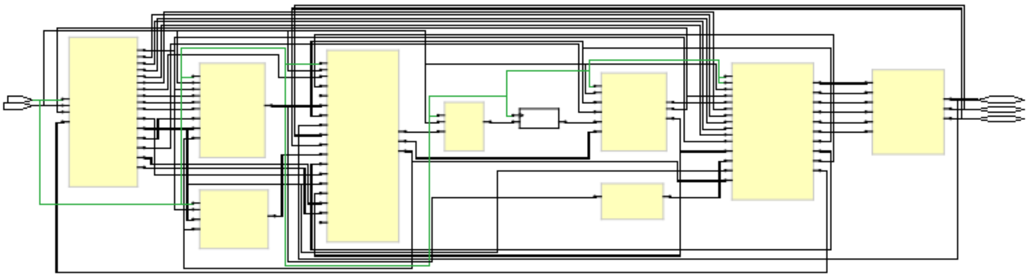


The following options are in the General panel.

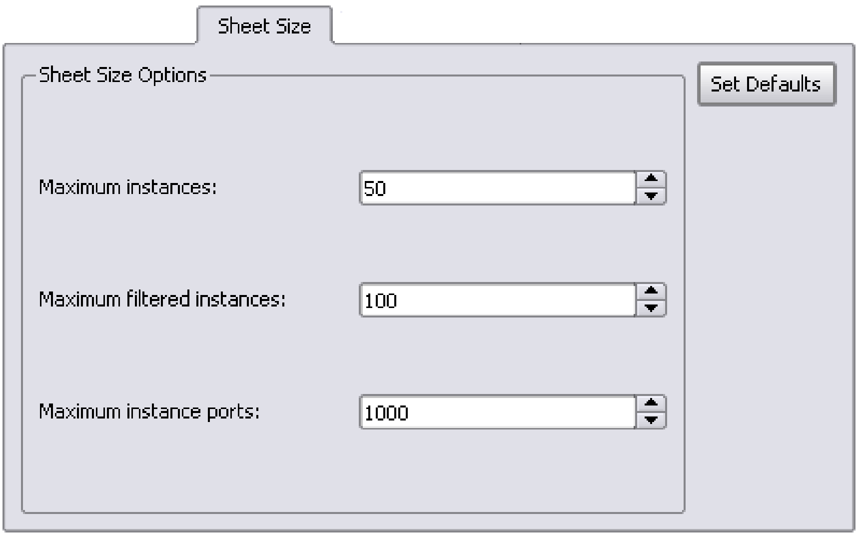
Field/Option	Description
Show hierarchy browser	When enabled, a hierarchy browser is present as the left pane of RTL and Technology views.
Show tooltip in schematic	When enabled, displays tooltips that hover objects as you move over them in the RTL and Technology schematic views.
Compact symbols	When 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 109.
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.
Dissolve levels	The number of levels to dissolve, during HDL Analyst->Dissolve Instances. See Dissolve Instances, on page 273
Instances added for expansion	The maximum number of instances to add during any operation (such as HDL Analyst->Hierarchical->Expand) that results in a *filtered* schematic. When this limit is reached, you are prompted to continue adding more instances.

Color-coded Clock Nets

Clock nets are displayed with the color green in the RTL and Technology views.



Sheet Size Panel

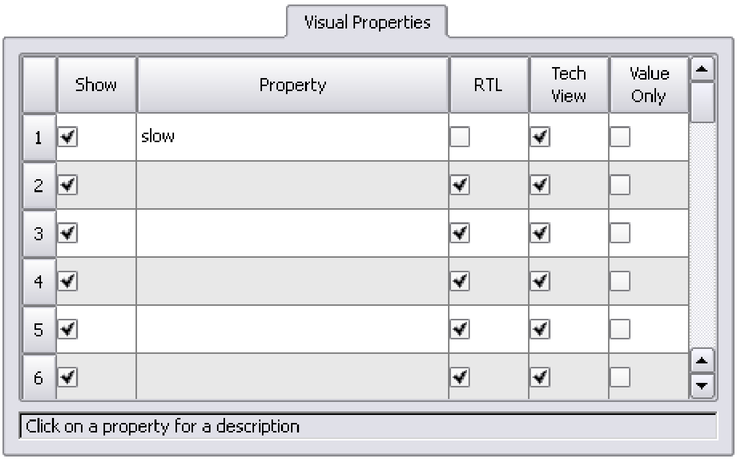


The following options are in the Sheet Size panel.

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 123.
Maximum filtered instances	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 *level*, when • the sheet itself is a level, and • each transparent instance is a level (even if inside another transparent instance). Whenever a given level has more child instances inside it than the value of Filtered Instances, it is divided into multiple sheets. (Only children are counted, not grandchildren or below. Instance A is a *child* of instance B if it is inside no other instance that is inside B.) 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. The Maximum filtered instances value must be at least the Maximum instances value. See Multiple-sheet Schematics, on page 123.
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 206 in the *User Guide*.

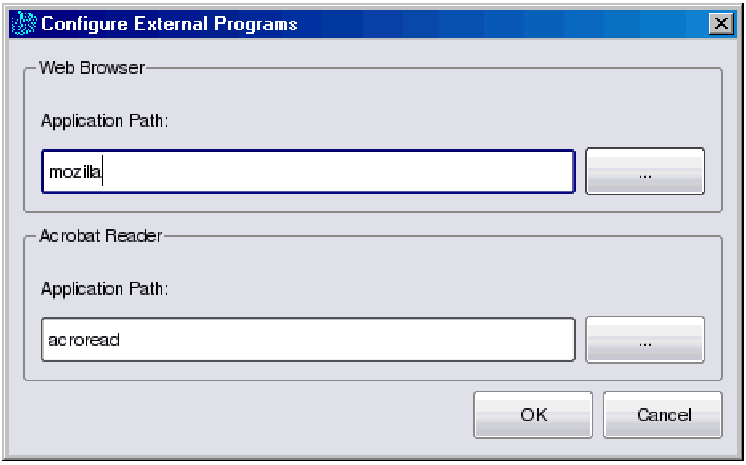


The following options are in the Visual Properties panel.

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 301 for details) and online documents.



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.

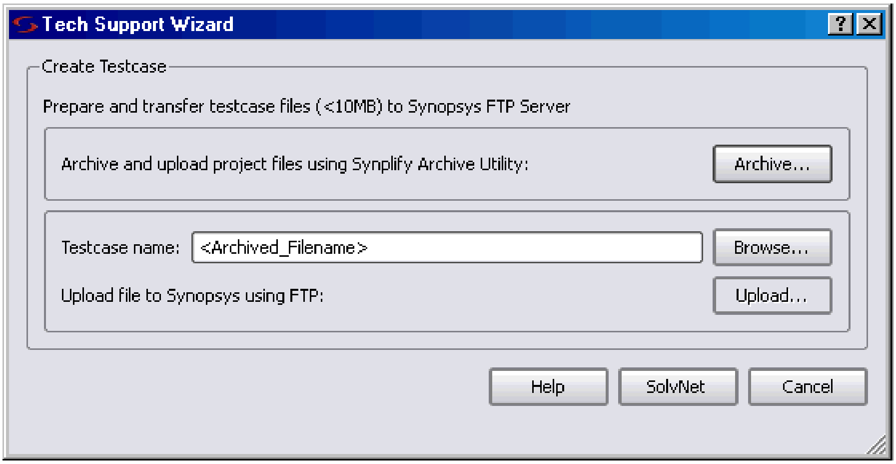
Tech-Support Menu

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

Command	Description
Submit Support Request	Opens the Technical Support wizard, which allows you to submit online support requests via SolvNet. The wizard includes provisions for attaching a testcase. See Submit Support Request Command, on page 296 for more information.
Web Support	Opens the Synopsys SolvNet Support page from where you can: • Log on to SolvNet to request Synopsys technical support. • Access the Synopsys Products, Downloads, Training, and Documentation pages that have links to product information. See Web Support Command, on page 299 for more information.

Submit Support Request Command

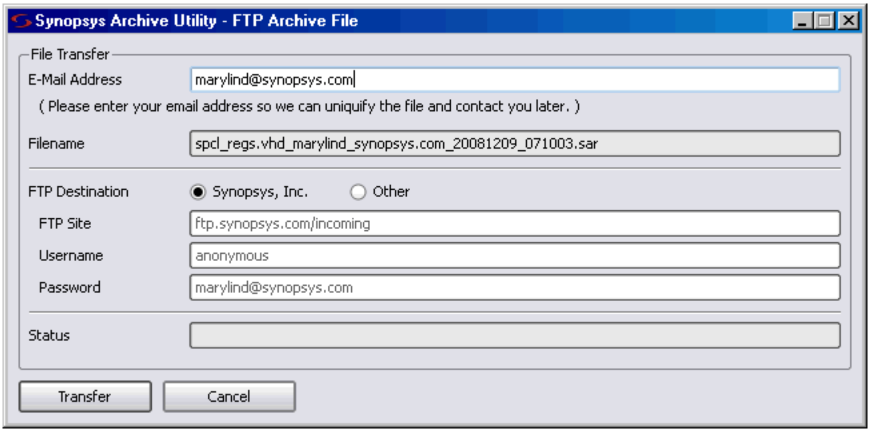
To open a request for Synopsys Technical Support, select Submit Support Request from the Tech-Support menu. This command brings up the web-based technical support wizard that helps you prepare the information required to provide technical support for your request through SolvNet.



Command	Description
Archive	Brings up the Synopsys Archive Utility to create an archive of your design. Note that designs are limited to 10 MBytes.
Testcase name	The name of the testcase or file to be transferred. To transfer multiple files, use the Synopsys Archive utility to create a single .sar file for the transfer.
Upload	Displays the FTP Archive File form to initiate the transfer of the testcase to the FTP file server. See FTP Archive File Form, on page 298.
SolvNet	Displays the Synopsys Sign In screen to access protected Synopsys applications. Signing in opens the SolvNet application which allows you to submit an online support request.

FTP Archive File Form

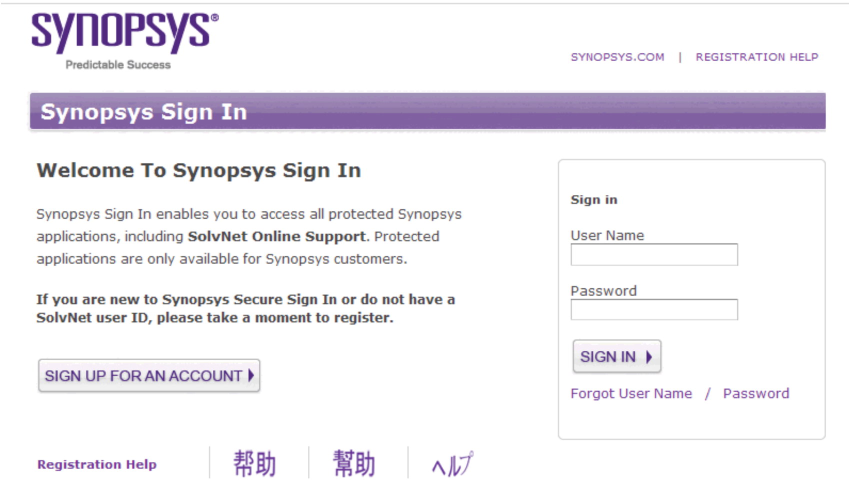
The FTP Archive File form is displayed when transferring a testcase or file to an FTP file server.



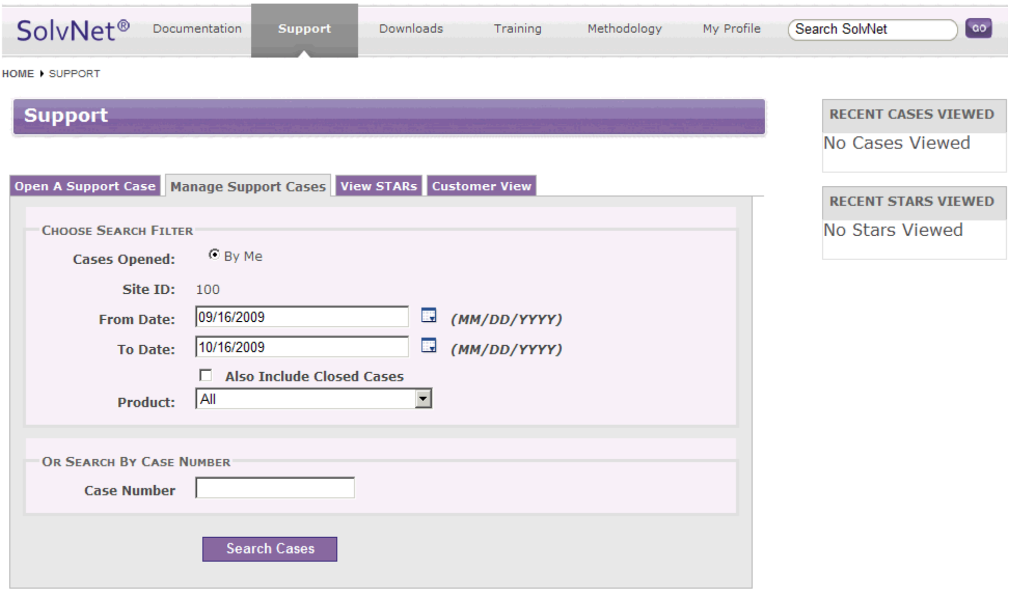
Command	Description
E-Mail Address	Your e-mail address. The name entered is used to create the FTP filename and is automatically entered into the Password field when using the Synopsys site.
Filename	A read-only field displaying the name of the testcase or file to be transferred. A .sar extension is added for single files (archive files are automatically assigned a .sar extension), and the filename is created with the e-mail address.
FTP Destination	Radio buttons that select either the Synopsys site or an alternate site. When Other is specified, you must supply FTP Site, Username, and Password entries.
Status	Reports the status of the transfer.
Transfer	Initiates the FTP file transfer. The results of the transfer are displayed in the Status field.

Web Support Command

Through the Synopsys SolvNet Support page, you can access Products, Downloads, Training, and Documentation pages on the Synopsys website as well as submit requests for technical support through SolvNet. First sign in to SolvNet.



Once you logon, the SolvNet Support page is displayed.



Web Menu

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

Command	Description
Go to SolvNet	Opens the home page for the Synopsys SolvNet Search support. This website contains links to useful technical information. You can search for new or updated articles or documentation, such as application notes, white papers, release notes, and other user-oriented documentation.
 Check Resource Center Messages	Opens a web page that contains updated messages, customized according to the options you set with Set Resource Center Options. You can also access this page by clicking the Message (envelope) icon in the status bar at the bottom of the application window. See Check Resource Center Messages Command, on page 302 for additional information.
Configure Resource Center	Lets you set options for Technical Resource Center (TRC) updates. See Configure Resource Center Command, on page 304 for details.
Go to Training Center	Opens the Synopsys training web page for Synopsys products. Synopsys offers both online web-based training courses, as well as classroom training courses taught by Synopsys personnel. Select the FPGA Implementation courses from the drop-down menu.
Synopsys Home	Opens the Synopsys home web page for Synopsys products.
FPGA Implementation Tools	Opens the Synopsys FPGA design solution web page for Synopsys FPGA products. You can find information about the full line of Synopsys FPGA Implementation products here.

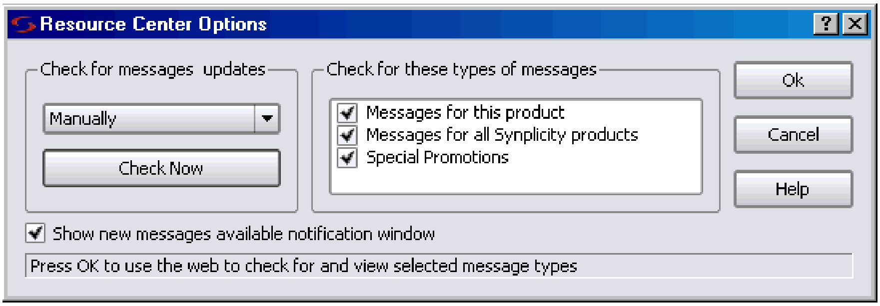
Check Resource Center Messages Command

This command lets you set options for messages from the Resource Center. SeUsing the Resource Center, on page 302 and Resource Center Messages, on page 303 for details.

Using the Resource Center

The following procedure explains how to set preferences and check the Technical Resource Center (TRC) for information.

1. To go to the TRC, select Web->Check Resource Center Messages.
2. To set preferences for accessing the Resource Center, select Web->Configure Resource Center and do the following:



– Set the frequency at which you wish to be notified.

– Select the type of information you want to receive.

– Select the Show new messages available notification window if you wish to receive immediate notification of any new messages.

If enabled, the following window appears when a new message is added.

- To go to the resource center immediately, click Check Now.

  If the TRC has never been configured, the envelope icon in the lower right of the application window appears yellow (to indicate available messages). Once you have configured TRC access, a yellow blinking envelope indicates new messages, and a gray envelope indicates there are no new messages.

3. To check updates, do one of the following:
   - Click the envelope icon in the status bar (lower right of the application window) when it's yellow and blinking to view new information.
   - Select Web->Check Resource Center Messages.

     If you have not set your TRC preferences, the software opens the Resource Center Options dialog box described in the previous step.
4. For other Synopsys FPGA product information, select the following:

Web->Go to SolvNet	http://solvnet.synopsys.com
Web->Go to Training Center	Synopsys training page
Web->Synopsys Home page	Synopsys home page
Web->FPGA Implementation Tools	Synopsys FPGA design solution information page
Tech-Support>Web Support	SolvNet technical support

Resource Center Messages

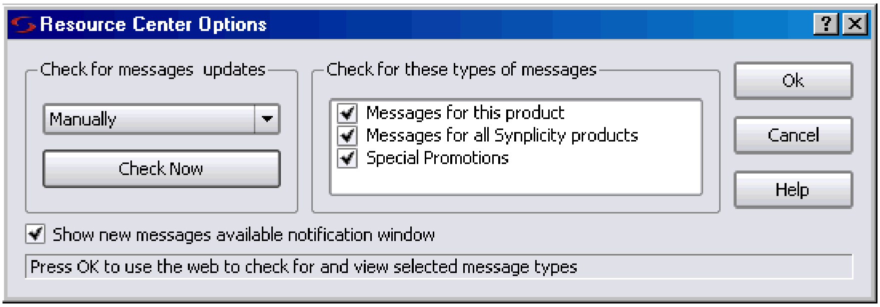
The envelope icon in the status bar (lower right of the application window) indicates when new messages are available:

Appearance	Signifies
Yellow or blinking yellow	New messages available
Gray	No new messages, or you have elected not to check for new messages.

Click on the icon to go to the Messages page of the Technical Resource Center.

Configure Resource Center Command

Sets options for automatic notification from the Technical Resource Center. From the Resource Center Options dialog box, you determine if you want to receive notification on updates for your product, all products, and/or promotional offers. You also determine how often you want the tool to check for new messages and alerts.



Option	Description
Check for messages and updates	Determines the frequency at which the software checks for updates. Select a setting from the menu: • Each invocation – checks for new messages and alerts each time you start the tool. • Weekly • Daily • Manually – does not automatically check for updates and/or alerts. With this setting, you must remember to go to the Resource Center to check for pertinent information on your product.
Check Now	Opens the home page for the Technical Resource Center (same as the Web->Check Resource Center Messages command).
Check for these types of messages	Determines the type of notification to receive. Check the boxes, as desired. • Messages for this product – Updates and critical bulletins for the product. • Messages for all Synplicity products – Updates and critical bulletins for all Synopsys FPGA products. • Special promotions – Promotional packages and pricing, occasional surveys, and other related information.

Help Menu

There are four help systems accessible from the Help menu:

- Help on the Synopsys FPGA synthesis tool (Help->Help)
- Help on standard Tcl commands (Help->TCL)
- Help on error messages (Help->Error Messages)
- Help on using online help (Help->How to Use Help)

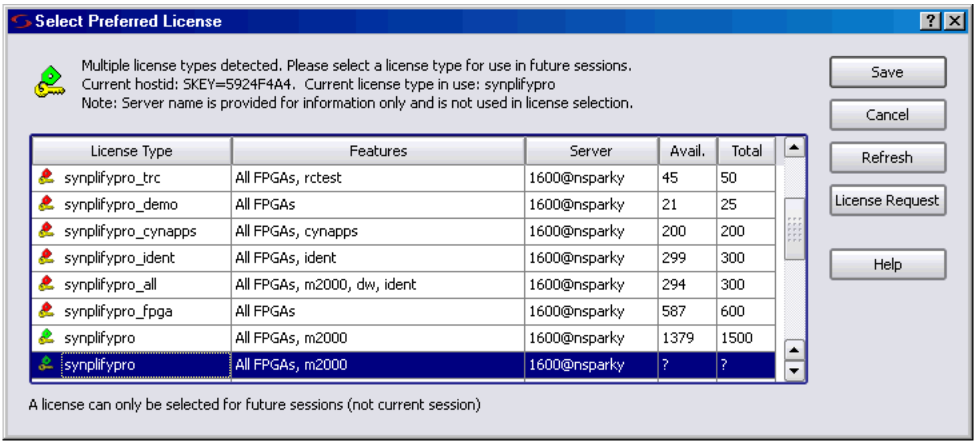
The following table describes the Help menu commands. Some commands are only available in certain views.

Command	Description
Help	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.
Online 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.
TCL	Displays help for Tcl commands.
Mouse Stroke Tutor	Displays the Mouse Stroke Tutor dialog box which provides information on the available mouse strokes – see Using Mouse Strokes, on page 79 for details.
License Agreement	Displays the Synopsys FPGA software license agreement.
License Request	Displays a dialog box where you can request a trial license or license upgrade.
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 306.
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 307.
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.

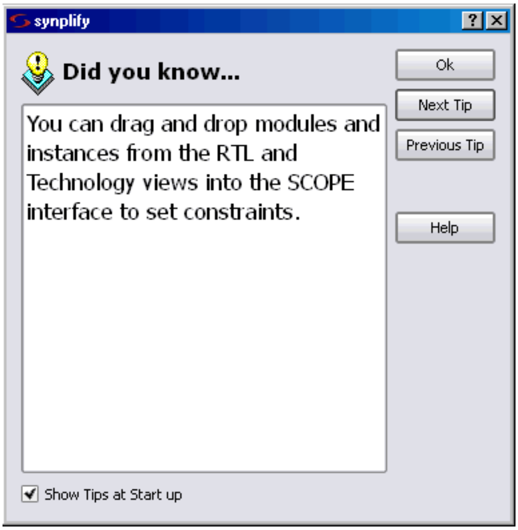
Preferred License Selection Command

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 FPGA synthesis 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

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 grayed 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 315 for details
SCOPE window	Contains commonly used commands from the Edit menu.
Watch Window	See Watch Window Popup Menu, on page 310 for details.
Tcl window	Contains commands from the Edit menu. For details, see Edit Menu Commands for the Text Editor, on page 152.
Text Editor window	See Text Editor Popup Menu, on page 311 for more information.
RTL and Technology views	See RTL and Technology Views Popup Menus, on page 329.
FSM viewer	See FSM Viewer Popup Menu, on page 313.

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 52.

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 152.

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 152 and HDL Analyst Menu, on page 264.

The following table lists the commands that are unique to the popup menu:

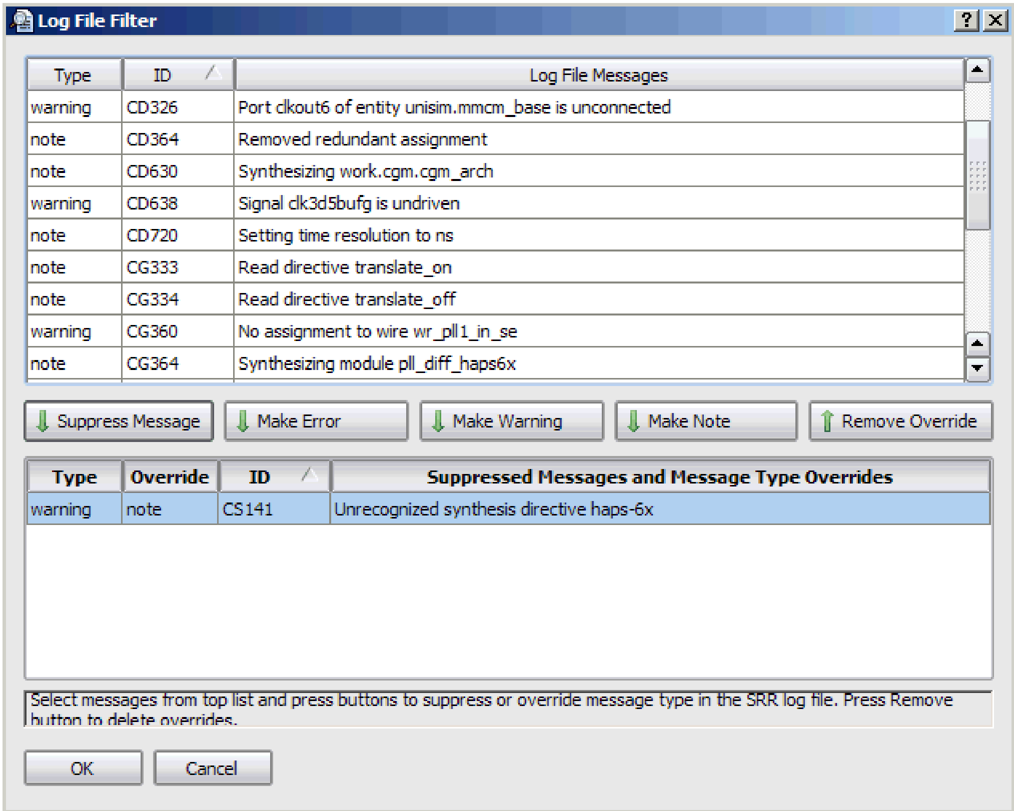
Command	Description
Filter Analyst	Filters your design to show only the currently selected objects in the HDL text file. This is the same as HDL Analyst->Filter Schematic.
Select in Analyst	Crossprobes from the Text Editor and selects the objects in the HDL Analyst view. To use this command, the Enhanced 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 196 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.

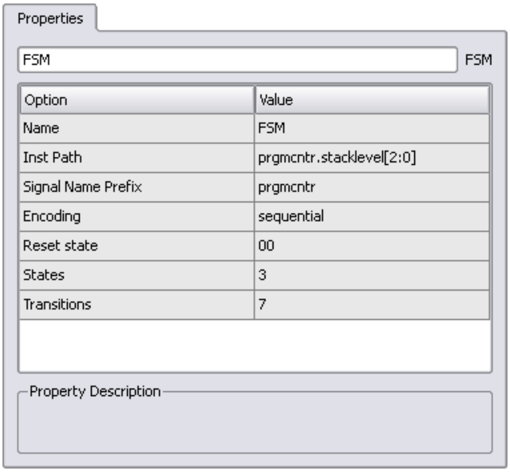
Function	Description
Make Warning button	Promotes the status of the selected note to a warning. The selected message is removed from the upper Log File Messages window and displayed in the lower window with the Override column indicating warning status.
Make Note button	Demotes the status of the selected 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

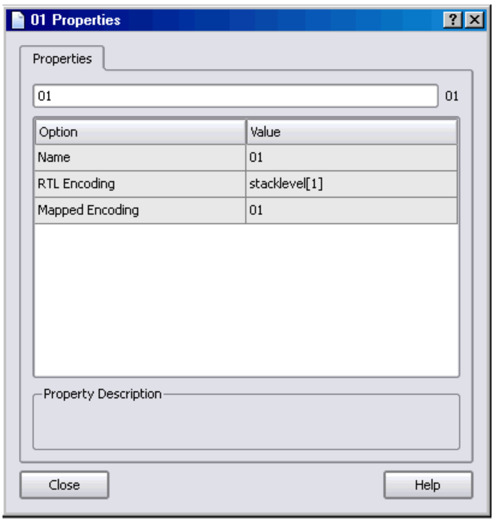
The popup menu in the FSM Viewer contains commands that determine what is shown in the FSM Viewer. The following table lists the popup commands in the FSM Viewer.

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 164.
Unfilter	See View Menu: FSM Viewer Commands, on page 164.
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.

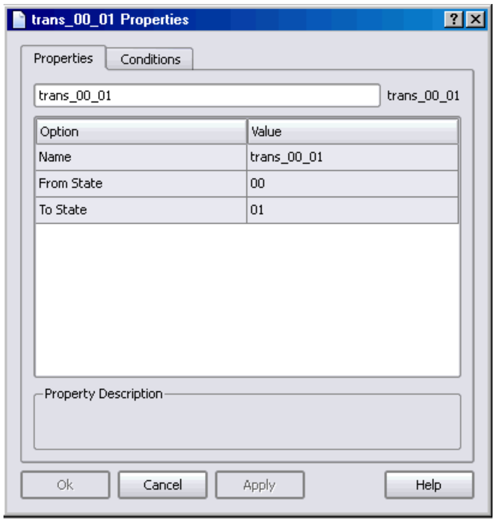
FSM Properties



State properties
(state selected)



Transition properties
(transition selected)



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 Commands

The following table lists the popup commands in the Synplify Pro Project Management view that are 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
Open Project	Displays the Open Project Dialog. See Open Project Command, on page 150.
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 280.
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 171.
New Implementation	Displays the Implementation Options dialog box. See Implementation Options Command, on page 183
Synthesize	Compiles and maps your design.
Compile Only	Compiles your design.

Command	Description
Write Output Netlist Only	Writes the mapped output netlist to structural Verilog (vm) or VHDL (vhm) format. Same as enabling: • Write Mapped Verilog Netlist • Write Mapped VDHL Netlist on the Implementation Results tab of the Implementation Options dialog box.
Arrange VHDL Files	Reorders the VHDL source files.
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 319.
Rename Folder	Renames an existing folder with the new name you specified in the Project view. See Rename Folder Command, on page 319.
Delete Folder	Deletes the specified folder and all its contents as necessary. See Delete Folder Command, on page 320.
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 321.
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 323.

Command	Description
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 173
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 321.
Open	Opens the file in the Text Editor.
Syntax Check	Runs a syntax check on your design code. Reports errors, warnings, or notes in the Tcl Window.
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 323.
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 173
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 183.
Change Implementation Name	Displays the Implementation Name dialog box, where you rename the selected implementation. (See Change Implementation Popup Menu Commands, on page 323.)
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 324.

Command	Description
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 323.)
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 325
Run	Starts a synthesis run on your design.
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 325.
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.
Project Management View -> Project FilesTab Popup Commands	

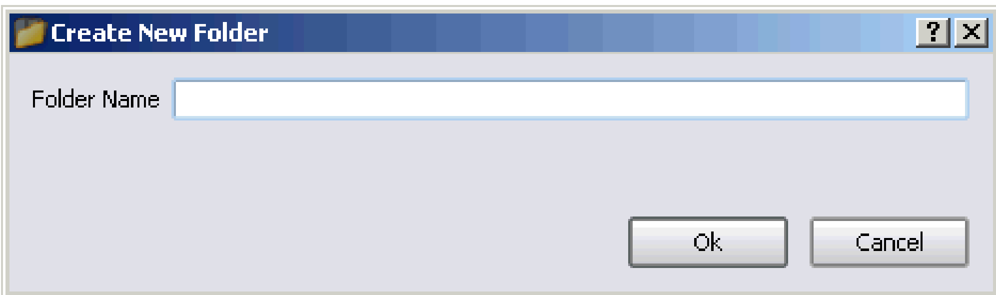
Command	Description
Project Options	With the project selected, displays project properties such as name and location. See Project Options Popup Menu Command, on page 324.
Show Compile Points	Displays the compile points of the selected implementation and lets you edit them. See Show Compile Points Popup Menu Command, on page 324.
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 327 for a description of the features.

Project Management View Popup Folder Commands

The Project view popup menu includes commands for manipulating folders.

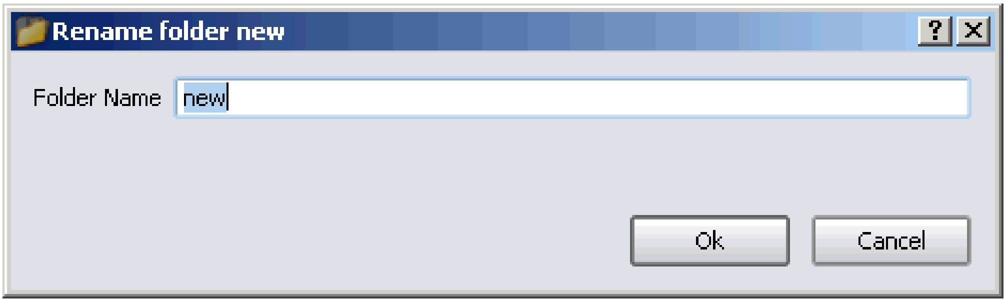
Add Folder Command

Use this option to add a folder to the Project view.



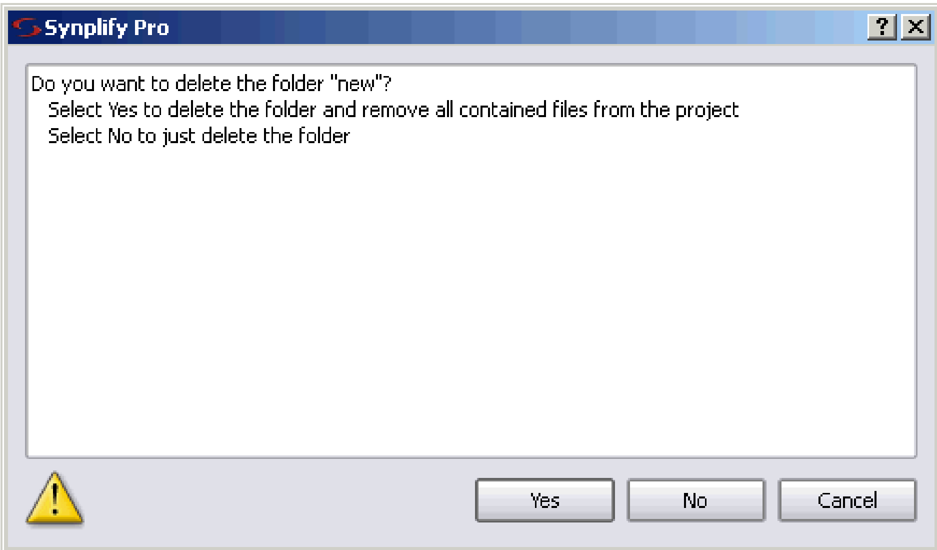
Rename Folder Command

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.

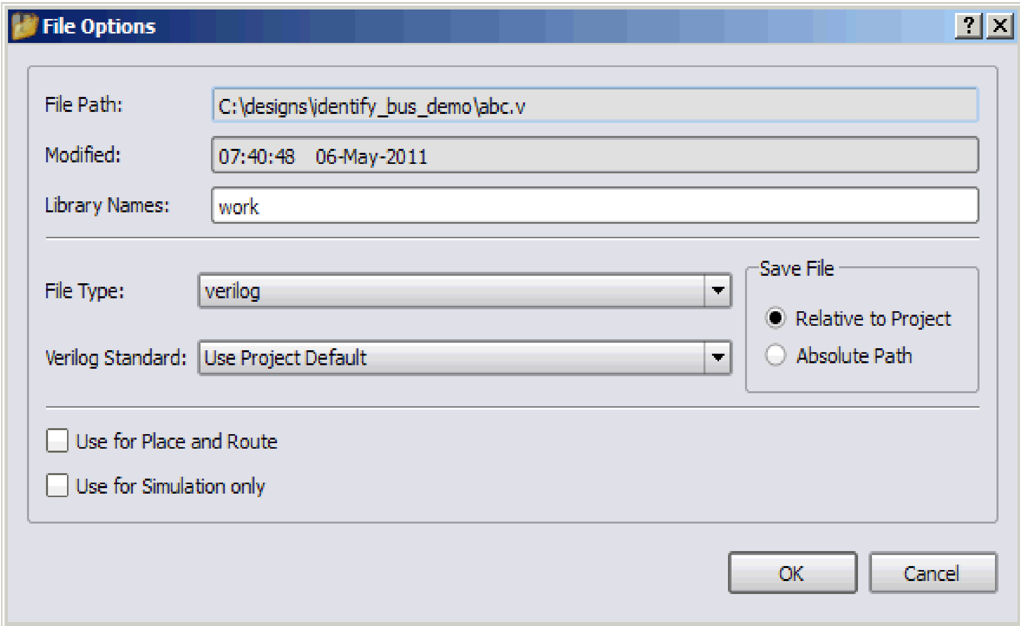
Feature	Description
Cancel	Select Cancel, to discontinue the operation.

File Options Popup Menu Command

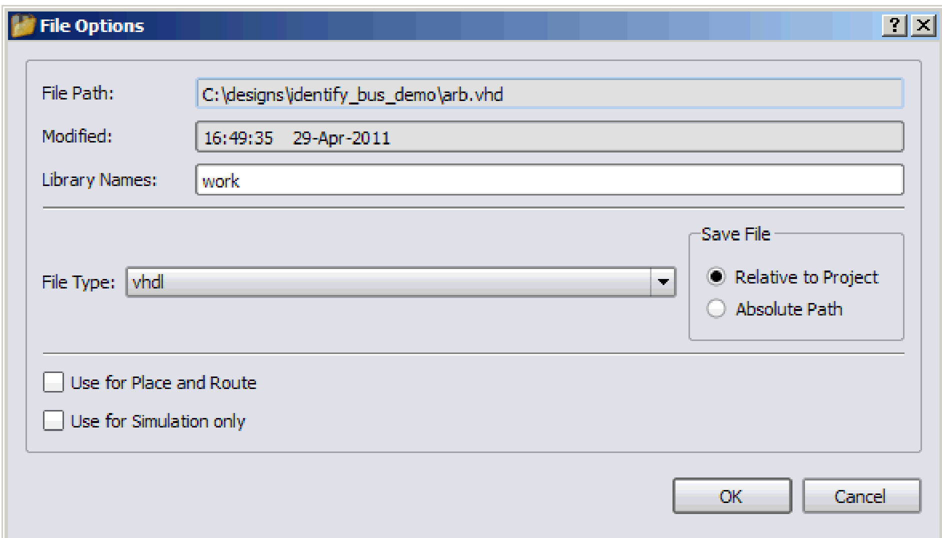
To display the File Options dialog box, right-click on a project file and select File Options from the popup menu. Specify the path as relative or absolute when listing the file in the project (`prj`) file and if the file is to be passed to the place-and-route tool.

Field/Option	Description
File Path	Path to the selected file.
File Type	The folder type for the selected file. You can select the file folder type from a large list of file types. Changing the folder file type does *not* 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.
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 174.
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)	Select the Verilog file type from the menu: Use Project Default, Verilog 95, Verilog 2001, or SystemVerilog. Use Project Default sets the type of the selected file to the default for the project (new projects default to SystemVerilog).
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.

The following is the Verilog dialog box:

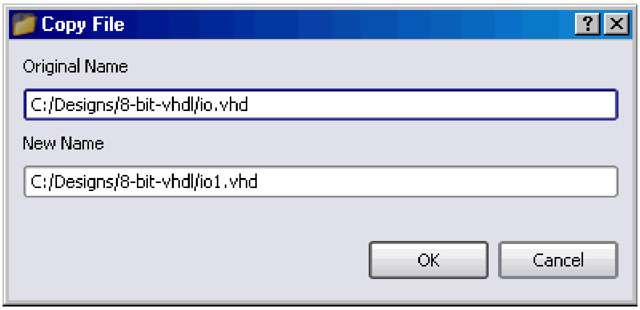


The following is the VHDL dialog box:



Copy File Popup Menu Command

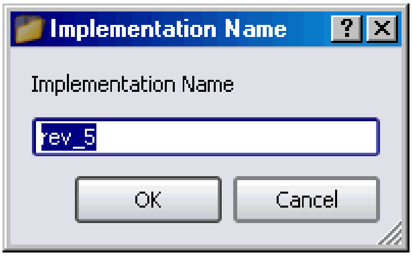
With a file selected, select the Copy File popup menu command to copy the selected file and add it to the current project. This displays the Copy File dialog box where you specify the name of the new file.



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.

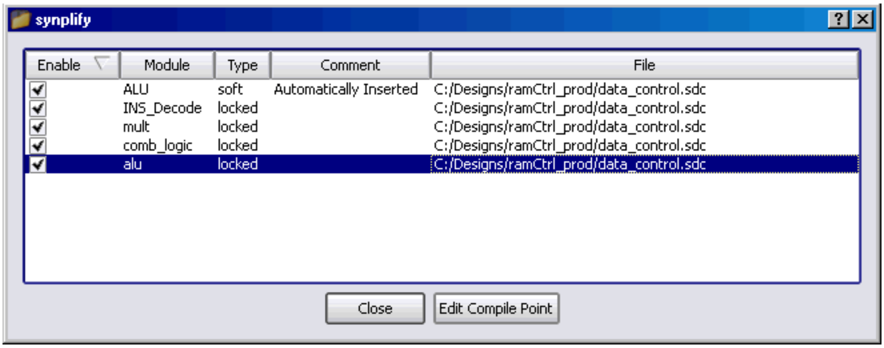
Command	Description
Change Implementation Name	The implementation name you specify is the new name for the implementation.
Copy Implementation	The currently selected implementation is copied and saved to the project with the new implementation name you specify.



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 365 and Synthesizing Compile Points, on page 378 of the *User Guide*.



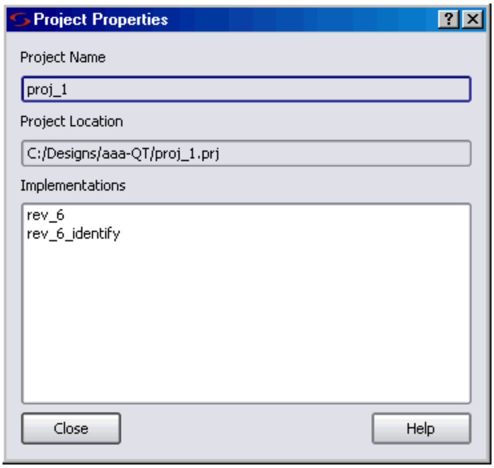
The columns Enb, Module, Type, and Comment in the dialog box correspond to the columns Enabled, Module, Type, and Comment in the SCOPE spreadsheet for the compile point. The File column lists the top-level constraint file where the compile point is defined.

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

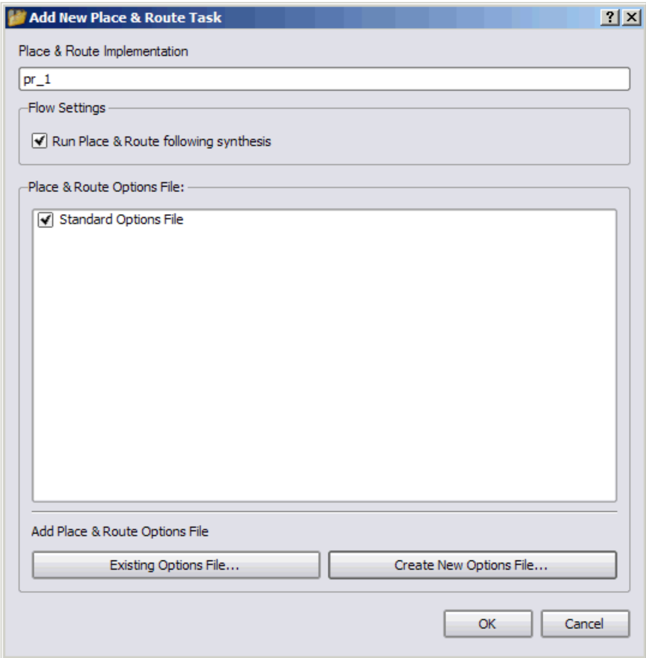
With a project selected, select the Project Options popup menu command to display the Project Properties dialog box and change the implementation of a project.

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.



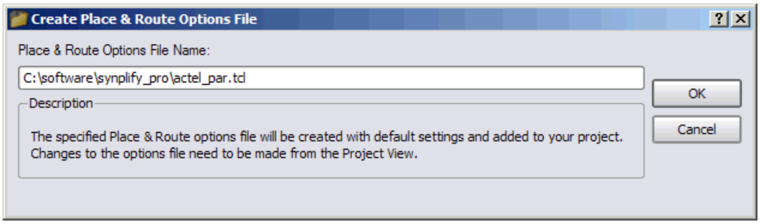
Add P&R Implementation Popup Menu Command

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 492 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.
Place & Route Options File	This option lets you specify a place & route options file. You can select either the: • Standard Options File – use this option to run the standard synthesis place-and-route flows. • Specify another option file.
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 492 for information about using this feature.

Command	Description
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 492 for information about creating a new options file.

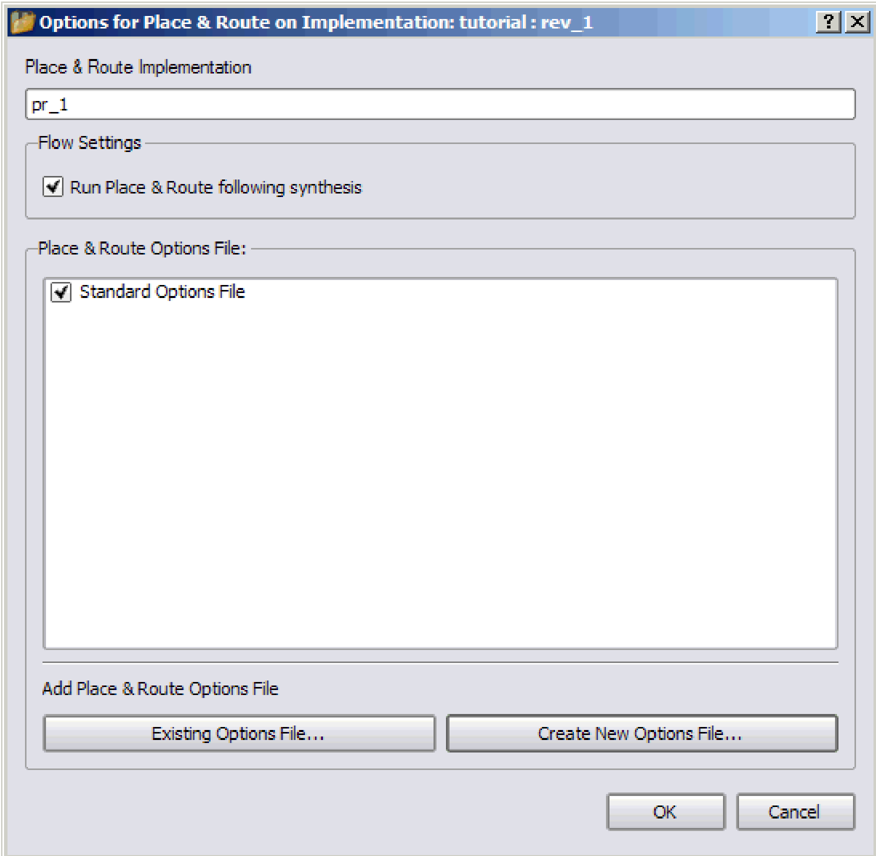


Once the par implementation is created, then you can right-click and perform any of the following options:

- P&R Options—See Options for Place & Route Jobs Popup Menu Command, on page 327.
- Add Place & Route Job—See Add P&R Implementation Popup Menu Command, on page 325.
- 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 325.


Options for Place & Route on Implementation: tutorial : rev_1
Place & Route Implementation
pr_1
Flow Settings
Run Place & Route following synthesis
Place & Route Options File:
Standard Options File
Add Place & Route Options File
Existing Options File...
Create New Options File...
OK
Cancel

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 329
- RTL View and Technology View Popup Menu Commands, on page 329

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 113.

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 264). The following table lists the unique commands.

Common Commands

Command	See...
Show Critical Path	HDL Analyst Menu: Timing Commands, on page 271
Timing Analyst	HDL Analyst Menu: Timing Commands, on page 271
Find	Find Command (HDL Analyst), on page 155
Filter Schematic	HDL Analyst Menu: Filtering and Flattening Commands, on page 267
Push/Pop Hierarchy	HDL Analyst Menu: RTL and Technology View Submenus, on page 264
Select All Schematic	HDL Analyst Menu: Selection Commands, on page 275
Select All Sheet	HDL Analyst Menu: Selection Commands, on page 275
Unselect All	HDL Analyst Menu: Selection Commands, on page 275
Flatten Schematic	HDL Analyst Menu: Filtering and Flattening Commands, on page 267
Unflatten Current Schematic	HDL Analyst Menu: Filtering and Flattening Commands, on page 267
HDL Analyst Options	HDL Analyst Options Command, on page 289
SCOPE->Edit Attributes (object <*name*>)	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 334), 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 <*module name*>)	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 334.
SCOPE->Edit Module Constraints (module <*module name*>)	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 272
Expand Paths	Hierarchical->Expand Paths, on page 266
Current Level Expand Paths	Current Level->Expand Paths, on page 267.
Show Context	Show Context, on page 272
Hide Instance	Hide Instances, on page 272
Unhide Instance	Unhide Instances, on page 272
Show All Hier Pins	Show All Hier Pins, on page 273
Dissolve Instance	Dissolve Instances, on page 273
Dissolve to Gates	Dissolve to Gates, on page 273

Port Selected

Command	**See...**
Expand to Register/Port	Hierarchical->Expand to Register/Port, on page 266
Expand Inwards	Hierarchical->Expand Inwards, on page 266
Current Level->Expand	Current Level->Expand, on page 266
Current Level->Expand to Register/Port	Current Level->Expand to Register/Port, on page 267

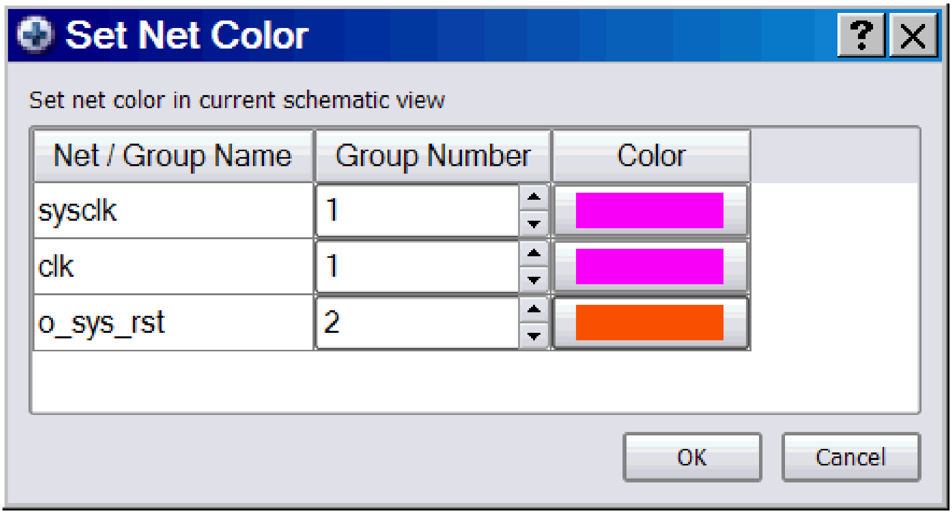
Current Level->Expand Paths	Current Level->Expand Paths, on page 267
Properties	Properties Popup Menu Command, on page 333

Net Selected

Command	**See...**
Goto Net Driver	Hierarchical->Goto Net Driver, on page 266
Select Net Driver	Hierarchical->Select Net Driver, on page 266
Select Net Instances	Hierarchical->Select Net Instances, on page 266
Current Level->Goto Net Driver	Current Level->Goto Net Driver, on page 267
Current Level->Select Net Driver	Current Level->Select Net Driver, on page 267
Current Level->Select Net Instances	Current Level->Select Net Instances, on page 267
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 332.

Set Net Color Popup Menu Command

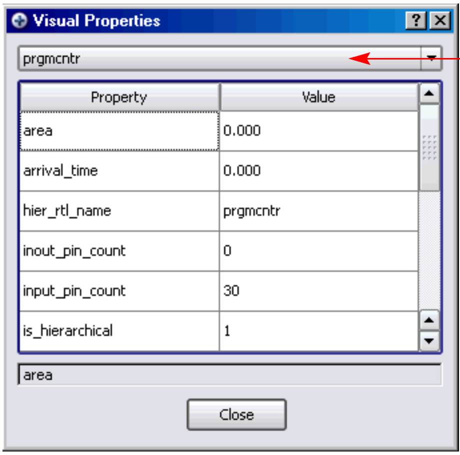
The set net color command sets the color of the selected net in the HDL Analyst for the current session. To use the command, select the desired net or nets in the RTL view and select set net color from the popup menu to display the dialog box.



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

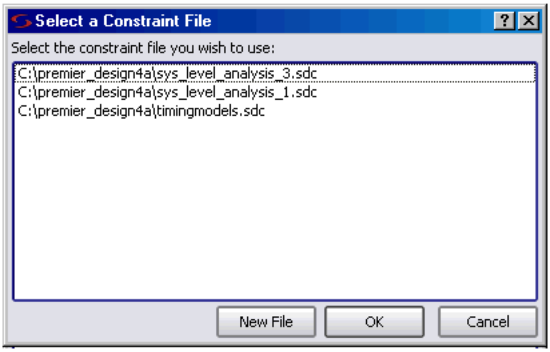
The software displays property information about the selected object when you right-click on a net, instance, pin, or port in a HDL Analyst view. See Visual Properties Panel, on page 294 or Viewing Object Properties, on page 206 in the *User Guide* for more information about viewing object properties.



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 Specifying Timing Exceptions, on page 126 of the *User Guide*.

Index

Symbols

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Z