

# **Synopsys FPGA Synthesis Synplify Pro Tutorial**

---

March 2010

<http://www.solvnet.com>

**SYNOPSYS®**

---

## Disclaimer of Warranty

Synopsys, Inc. makes no representations or warranties, either expressed or implied, by or with respect to anything in this manual, and shall not be liable for any implied warranties of merchantability or fitness for a particular purpose of for any indirect, special or consequential damages.

## Copyright Notice

Copyright © 2010 Synopsys, Inc. All Rights Reserved.

Synopsys software products contain certain confidential information of Synopsys, Inc. Use of this copyright notice is precautionary and does not imply publication or disclosure. No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language in any form by any means without the prior written permission of Synopsys, Inc. While every precaution has been taken in the preparation of this book, Synopsys, Inc. assumes no responsibility for errors or omissions. This publication and the features described herein are subject to change without notice.

## Trademarks

### Registered Trademarks (®)

Synopsys, AMPS, Astro, Behavior Extracting Synthesis Technology, Cadabra, CATS, Certify, Design Compiler, DesignWare, Formality, HDL Analyst, HSPICE, Identify, iN-Phase, Leda, MAST, ModelTools, NanoSim, OpenVera, PathMill, Physical Compiler, PrimeTime, SiVL, SCOPE, Simply Better Results, SNUG, SolvNet, Synplicity, the Synplicity logo, Synplify, Synplify Pro, Synthesis Constraints Optimization Environment, TetraMAX, VCS, Vera, and YIELDirector are registered trademarks of Synopsys, Inc.

### Trademarks (™)

AFGen, Apollo, Astro-Rail, Astro-Xtalk, Aurora, AvanWaves, BEST, Columbia, Columbia-CE, Confirma, Cosmos, CosmosLE, CosmosScope, CRITIC, DC Expert, DC Professional, DC Ultra, Design Analyzer, Design Vision, DesignerHDL, DesignPower, Direct Silicon Access, Discovery, Eclipse, Encore, EPIC, Galaxy, HANEX, HAPS, HapsTrak, HDL Compiler, Hercules, Hierar-

---

chical Optimization Technology, High-performance ASIC Prototyping System, HSIM, HSIM<sup>plus</sup>, i-Virtual Stepper, IICE, in-Sync, iN-Tandem, Jupiter, Jupiter-DP, JupiterXT, JupiterXT-ASIC, Liberty, Libra-Passport, Library Compiler, Magellan, Mars, Mars-Rail, Mars-Xtalk, Milkyway, ModelSource, Module Compiler, MultiPoint, Physical Analyst, Planet, Planet-PL, Polaris, Power Compiler, Raphael, Saturn, Scirocco, Scirocco-i, Star-RCXT, Star-SimXT, System Compiler, System Designer, Taurus, TotalRecall, TSUPREM-4, VCS Express, VCSi, VHDL Compiler, VirSim, and VMC 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.

## **Restricted Rights Legend**

Government Users: Use, reproduction, release, modification, or disclosure of this commercial computer software, or of any related documentation of any kind, is restricted in accordance with FAR 12.212 and DFARS 227.7202, and further restricted by the Synopsys Software License and Maintenance Agreement. Synopsys, Inc., Synplicity Business Group, 600 West California Avenue, Sunnyvale, CA 94086, U. S. A.

Printed in the U.S.A  
March 2010



# Contents

---

Introduction to the Tutorial .....	8
Start the Software .....	8
Download Tutorial Files .....	10
Tutorial Directory Structure .....	11
Synthesis Files .....	12
See Also .....	14
Tutorial Design Flow .....	15
Create Project .....	16
Setup Project and Add Design Files .....	16
Compile Design and Check Log File .....	22
Additional Analysis after Compile .....	25
Setup Implementation for Synthesis .....	26
Set Constraints .....	26
Set Device Options .....	30
Run Logic Synthesis .....	39
Analyze Logic Synthesis Results .....	40
Examine the Technology View .....	40
Check Timing .....	42
Analyze Critical Paths in the Technology View .....	45
Set Additional Constraint and Resynthesize .....	46
Check Results .....	49
<i>Appendix A: Early Analysis (Compile Phase)</i> .....	50
Analyze Compile Results (RTL) and Navigate Hierarchy .....	50
Find and Crossprobe .....	55
Filter, Expand, Hide, and Dissolve .....	61

---

# Synplify Pro Tutorial

---

The tutorial shows you how to use the Synplify Pro software in the FPGA logic design process. Information is organized into these topics:

- [Introduction to the Tutorial](#), on page 8
- [Tutorial Design Flow](#), on page 15
- [Create Project](#), on page 16
- [Setup Implementation for Synthesis](#), on page 26
- [Run Logic Synthesis](#), on page 39
- [Analyze Logic Synthesis Results](#), on page 40
- [Improve Results](#), on page 46
- [Appendix A: Early Analysis \(Compile Phase\)](#), on page 50

# Introduction to the Tutorial

The tutorial is designed to walk you through the Synplify Pro design flow using some typical tasks and familiarize you with the user interface.

The tutorial design is an 8-bit micro controller. After completing the tutorial, you will be familiar with the tool and able to apply the knowledge you gained to your own, more complicated designs.

The tutorial assumes that you have

- Installed the software correctly and obtained the necessary licenses.
- Basic understanding of logic synthesis using the Synopsys FPGA tools.

The remaining sections include the following topics:

- [Start the Software, on page 8.](#)
- [Download Tutorial Files, on page 10.](#)
- [Tutorial Directory Structure, on page 11.](#)
- [Synthesis Files, on page 12](#)
- [See Also, on page 14](#)

## Start the Software

You can start the software and run the tutorial from a Windows or Linux workstation.

1. On Windows, choose the current release of the software from:  
Start->Programs->Synopsys->FPGA Synthesis D-2010.03->Synplify Pro.

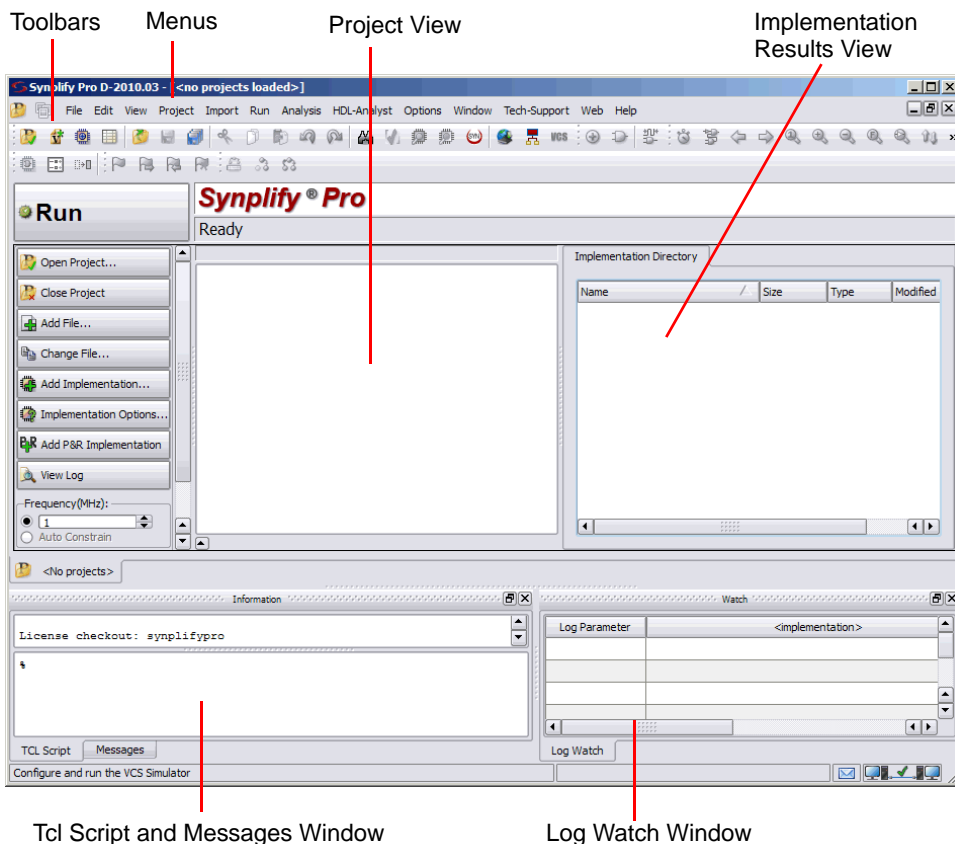
If you use any other version of the software, results may not exactly match the results in the tutorial, although you can still follow the general methodology described in this document.

2. On Linux, type this at the command line:

```
synplify_pro
```

The command starts the synthesis tool. If you have run the software before, the window displays the previous project.





If you do not see the Tcl Script and Messages window and Log Watch window, select View->TCL Window and View->Watch Window or View->Output Window. For your information, you can access commands in different ways: through the main menu, popup menus, keyboard shortcuts, and icons. The tutorial uses different methods to access the commands. For more information about the interface, see the *Synopsys FPGA Synthesis Reference Manual*.

## Download Tutorial Files

You can download the tutorial design files and the tutorial instructions from the Synopsys SolvNet® website.

1. Logon to SolvNet.
2. Select the applicable release for the tutorial and download the platform-specific version of the design files.
3. Unzip the tutorial files.
  - On Windows, use Winzip to extract the tutorial files.
  - On Linux, type the following at the command line:  

```
gunzip tutorial.tar.gz
```

Then, to extract the tutorial files, type the following at the command line:

```
tar -xvf tutorial.tar
```
  - Open or print the tutorial instructions (tutorial.pdf) from the SolvNet when you are ready to begin the tutorial.
4. Copy the tutorial directory to your working area. Keep the directory structure, because the tutorial is based on this structure. Refer to [Tutorial Directory Structure, on page 11](#). When you work on your own designs, you can set up the structure as you want.
5. Make sure you have read and write privileges for the tutorial files.

## Tutorial Directory Structure

The input files for this tutorial are provided in the Synplify Pro tutorial directory after you download files from the Synopsys SolvNet website and setup your project. The project and constraint files will be created using this tutorial. However, if you prefer you can use these files provided with the design as well.

---

**Note:** This directory structure is used for the tutorial because it reflects the way the tool structures the files in the Project view. However, when you run the Synplify Pro software on your own, you can create whatever directory structure works best for your design.

---

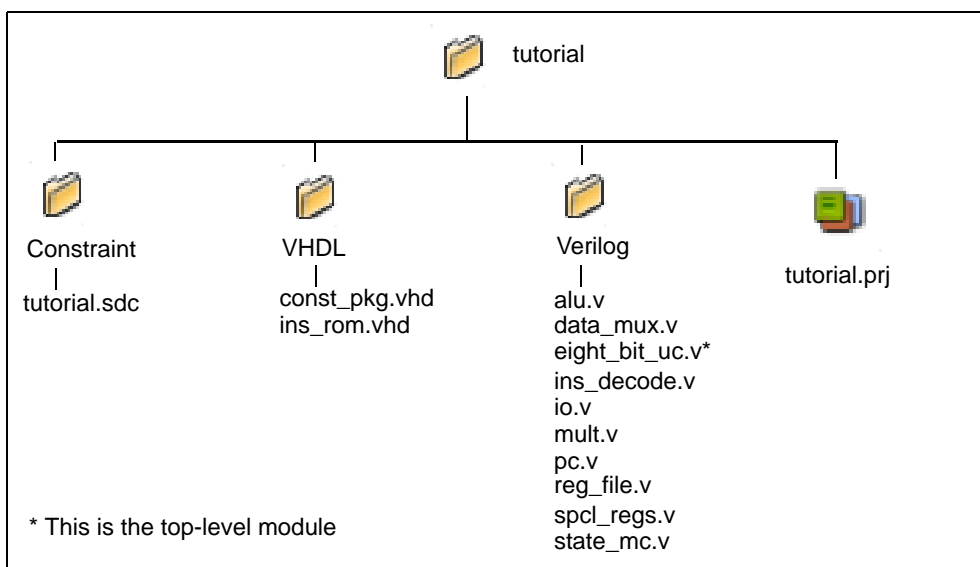


Figure 1: Synplify Pro FPGA tutorial Directory Structure

For descriptions of these files see [Input Files, on page 12](#).

Figure 2 shows the directory structure for the implementation results files, using the default implementation name rev\_1. You specify the location of the results directory when you set implementation options for your project.

After you complete this exercise, the results directory typically contains the file types shown below.

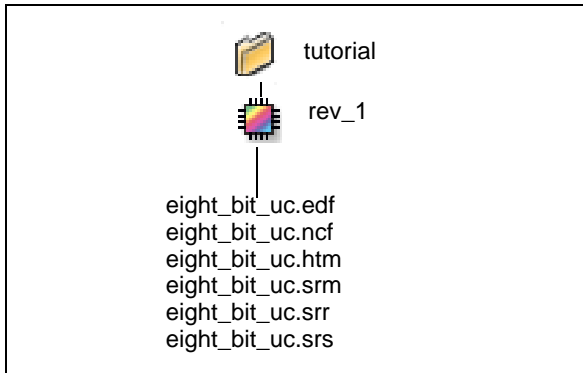


Figure 1: Results Directory Structure

For descriptions of these files, see [Output Files, on page 13](#).

## Synthesis Files

This section briefly describes the files required to run synthesis and the files generated during synthesis that are output to the user-specified implementation results directory.

### Input Files

Here is a brief description of the input files:

- `.v/.vhd`—contains the HDL source files. The HDL source files can also contain a mixture of VHDL and Verilog source files. `eight_bit_uc.v` is the top-level module.
- `constraint/tutorial.sdc`—user-specified constraint file, contains the timing constraints

The constraint file will be created using this tutorial. However, you can use the `.sdc` file provided with the design, if preferred.

- `tutorial.prj`—tutorial project file, contains all the information required to complete a design. This file contains references to source files, and specifications for the target device.

The project file will be created using this tutorial.

## Output Files

Here is a brief description of the files that are typically output to the Implementation Results directory:

- `.edf`—Xilinx design netlist in the format of the supported target place-and-route tool
- `.htm`—HTML format of the log file containing the synthesis results. See the `.srr` file below for a description of its contents.
- `.ncf`—Xilinx netlist constraint file; contains all of the constraints for the design
- `.srm`—output by the mapper stage of the process, contains the actual technology-specific mapped design. This is the representation displayed through the technology view in HDL Analyst.
- `.srr`—text format of the log file containing the synthesis results. The `project_name.srr` file contains all warnings and errors encountered during synthesis as well as performance information such as clock frequency, critical paths and run times. There is also information on area, cell usage and FSM extraction. To view this file, click on the View Log button in the Project view.
- `.srs`—output by the compiler stage of the process, contains the RTL level (schematic) view of the design. This is the representation displayed through the RTL view in HDL Analyst.

---

**Note:** You can delete or rename the `tutorial.sdc` files in this directory if you want to create your own as part of the tutorial exercises.

---

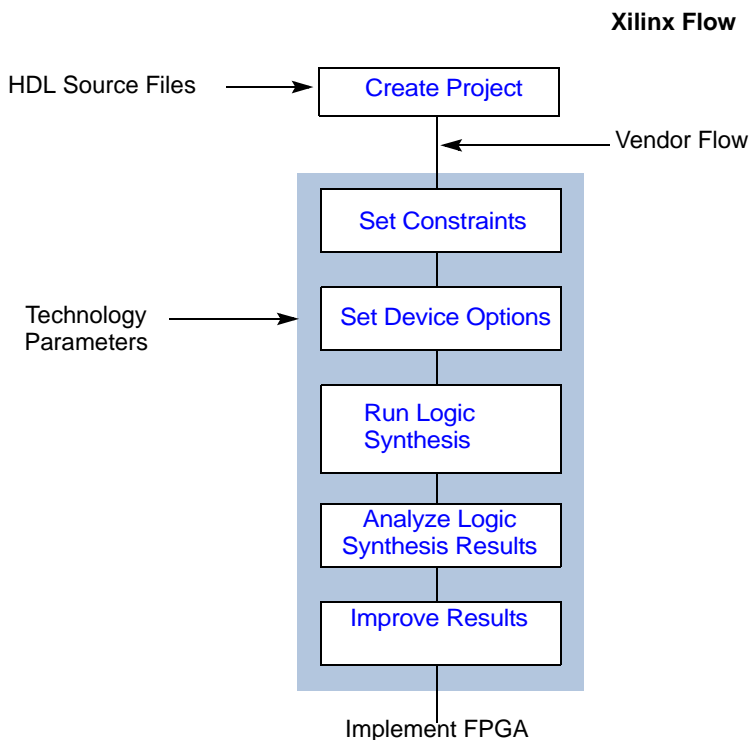
## See Also

The tutorial does not cover all the possible tasks you could do. For additional information, refer to the following sources:

<b>For information about...</b>	<b>See the...</b>
Installation	Installation instructions on SolvNet
Common tasks not covered in the tutorial	<i>Synopsys FPGA Synthesis User Guide</i>
Advanced techniques	<i>Synopsys FPGA Synthesis User Guide</i>
User interface descriptions	<i>Synopsys FPGA Synthesis Reference Manual</i>
Commands and syntax	<i>Synopsys FPGA Synthesis Reference Manual</i>

# Tutorial Design Flow

This flow diagram graphically illustrates the procedures in this tutorial:



The remaining sections describe how to complete the tasks for this flow.

# Create Project

The first step is to set up your project. A project is a file that defines the HDL source files, implementation files and device option settings. This section shows you how to set up a project file, handle messages, and do some typical analysis operations with the HDL Analyst tool. This project information is organized as follows:

- [Setup Project and Add Design Files](#)
- [Compile Design and Check Log File](#)
- [Additional Analysis after Compile](#)

## Setup Project and Add Design Files

To run synthesis, you need a project file. A project contains the data needed for a particular design: the source files, the name of the synthesis results file, and your device option settings. The following procedure shows you how to set up a project file.

1. In the project window, select File->Build Project to open the Select Files to Add to Project dialog box. Navigate to your source files by selecting the *install\_dir/tutorial* directory.



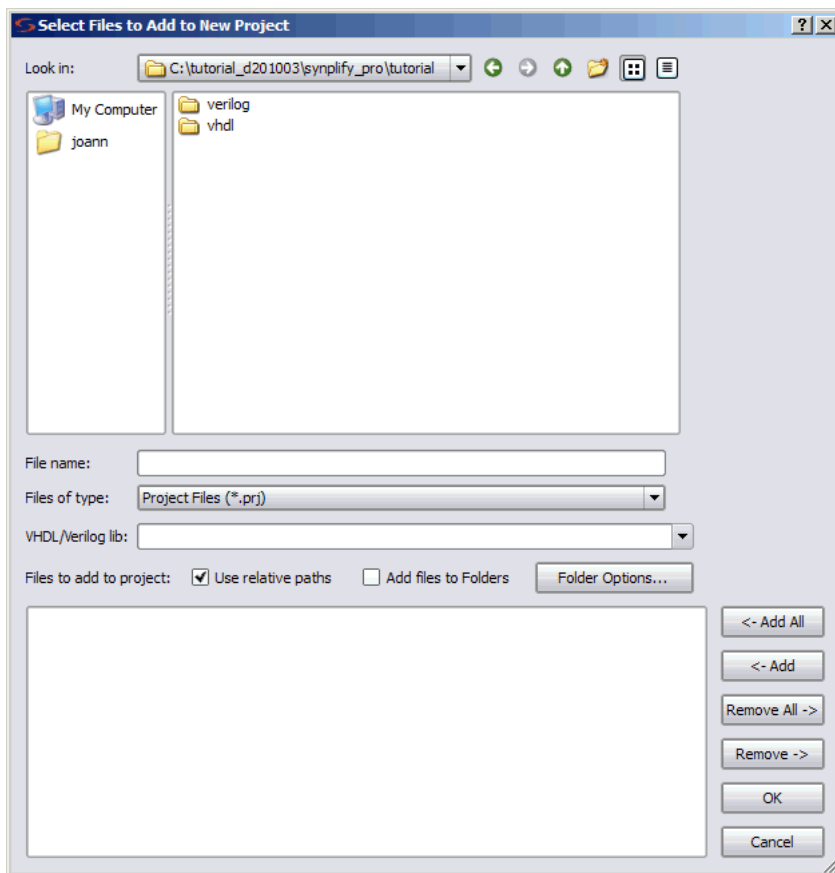


Figure 2: Select Files to Add to Project Dialog Box

2. Open the vhd folder. Make sure Files of type field is showing either All Files (\*) or VHDL Files (\*.vhd).

For this exercise, add both VHDL files in the folder. Click on the <-Add All button, then click OK.

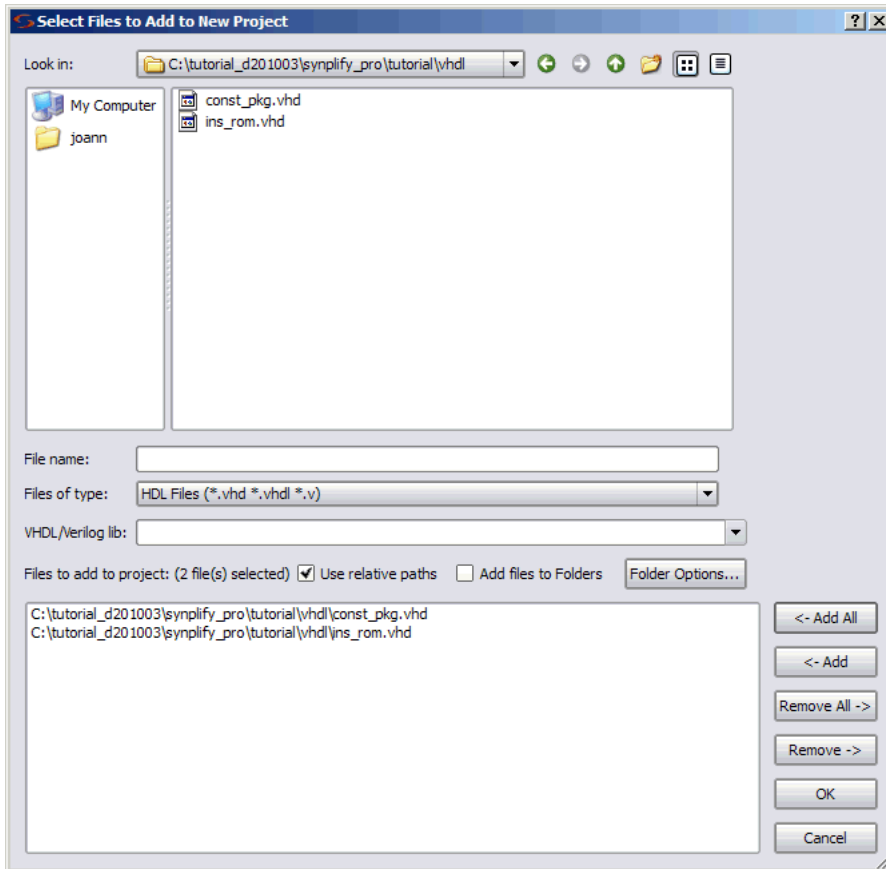


Figure 3: Add VHDL Files

- After clicking OK to add the files, you will return to the Project view.

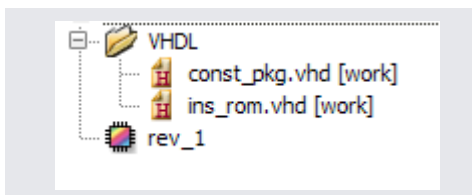


Figure 4: Newly Created Project File

Your project window displays a newly-created project file called `const_pkg` (the first source file in the list) with a folder called `VHDL` and a directory under it called `rev_1`, which represents the first implementation of your design.

3. Set the source file hierarchy, if necessary.

- If you do not see a `VHDL` folder under the `const_pkg` directory, set this preference by selecting the `Options->Project View Options` command and enabling the `View Project Files in Type Folders` check box. This separates one kind of file from another in the Project view by putting them in separate folders.

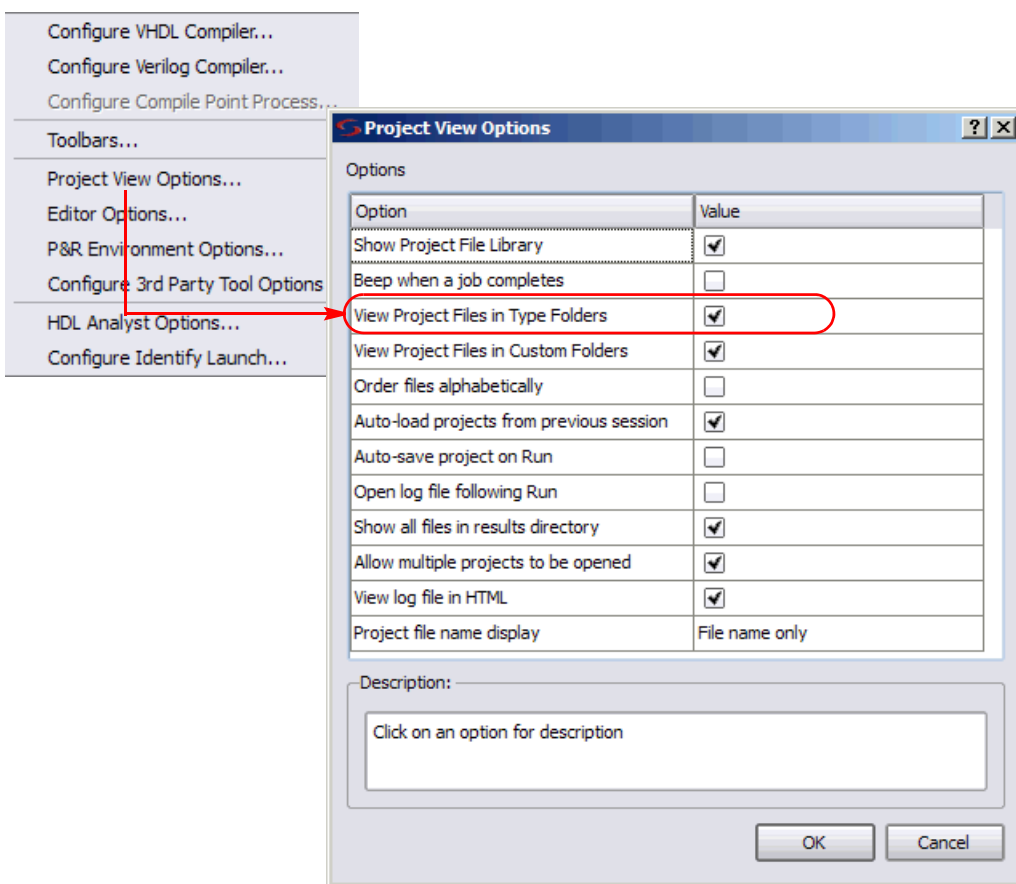


Figure 5: Select Project View Options->View Project Files in Type Folders

- Click on the Add File button in the Project view and change to the verilog directory. Make sure Files of type field is showing either All Files (\*) or Verilog Files (\*.v).
- For this exercise, add all the Verilog files in the folder. Click on the <-Add All button, then click OK.

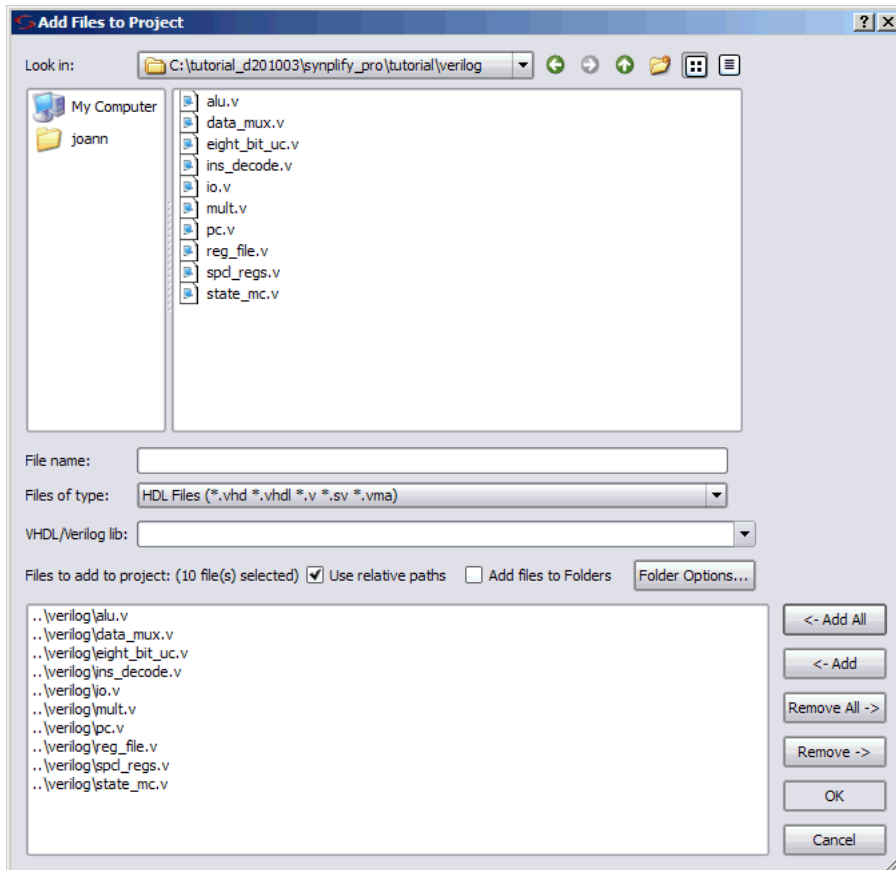


Figure 6: Add Verilog Files

A virtual design directory structure is created in the Project view. Figure 7 shows the UI after adding the design files.

- Click on the plus sign next to the Verilog folder to see a list of the Verilog source files that you added for the project.

- Drag the top-level Verilog file to the last position in the list of files to manually rearrange them.
- The project window reflects your changes. Make sure that the top-level file (`eight_bit_uc.v`) is the last in the list of files in the Project window. The order of the remaining files does not matter.

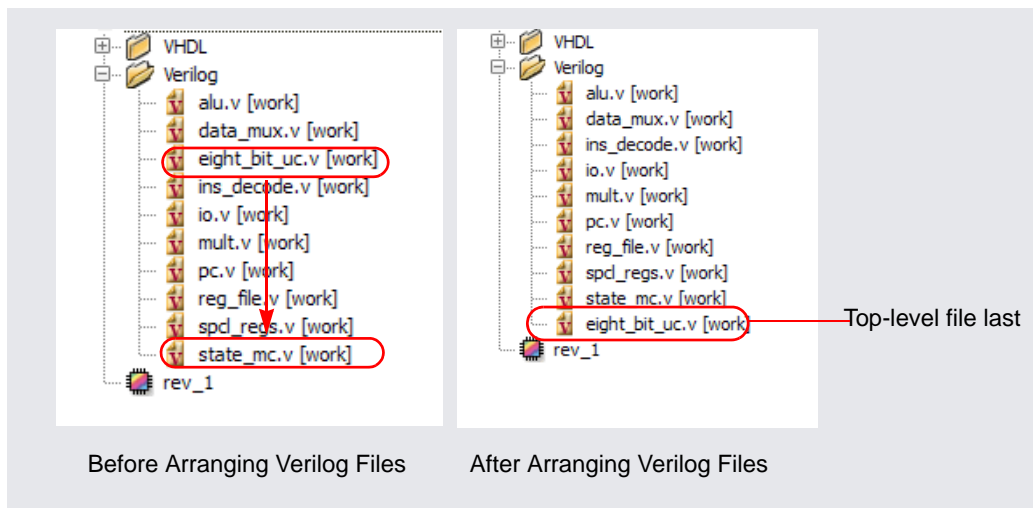
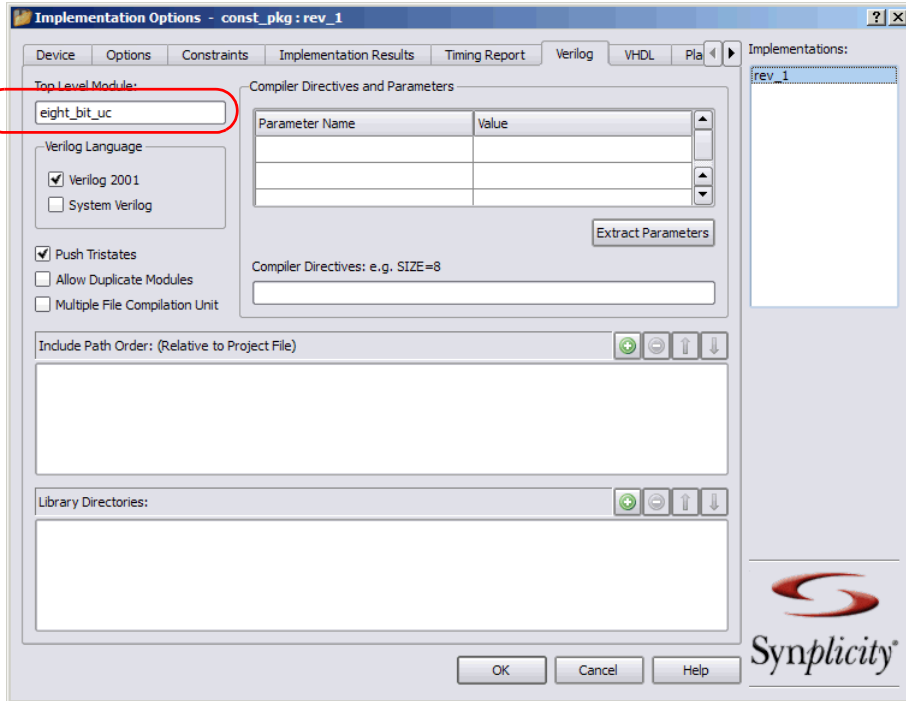


Figure 7: Verilog Files Before and After Arranging

However, since we are using a mixed-language design for this exercise, and the top-level module is Verilog, specify the top-level module in the Verilog panel of the Implementation Options dialog box.

6. Click on the Impl Options button and click on the Verilog tab. See [Set Device Options, on page 30](#) for more information.



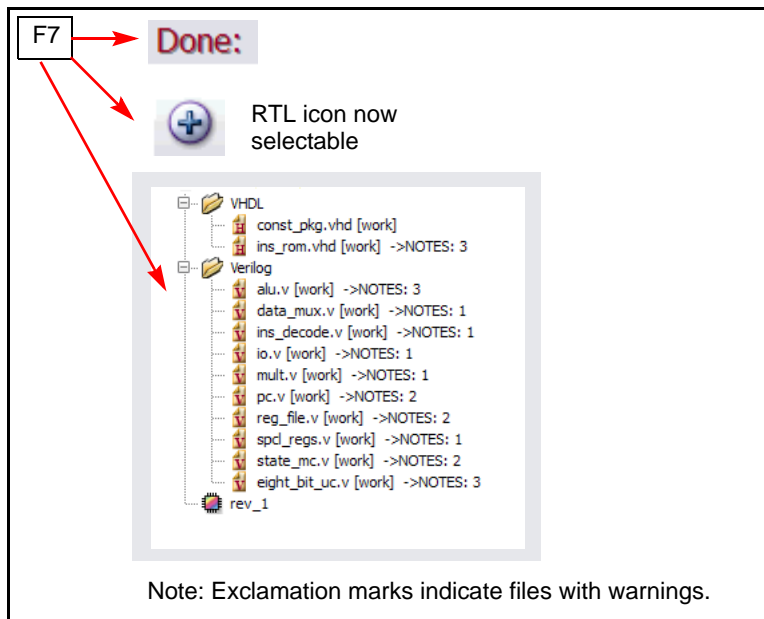
7. Then select File->Save As, move up a directory level and type tutorial for the name of the project, and click Save.

## Compile Design and Check Log File

For the purposes of this tutorial, you compile and check the log file as separate steps. Normally, compilation is part of synthesis when you press Run.

1. Press F7 or select Run -> Compile Only.

The software optimizes the logic for your design using technology-independent operations, and checks for syntax and hardware-related synthesis errors. When it has compiled, you see the following changes in the Project view:



## 2. Review the messages in the Message viewer.

- Click the Messages tab in the Tcl window. Enlarge the message window if needed, or drag it out of its docked location to get a full-size window. You see all the messages listed. Notes have an 'n' icon, and warnings have a yellow triangle with an exclamation mark.
- Locate the Found RAM mem\_regfile, depth=32, width=8 entry and click the ID number CL134.

The online messages help reports that the compiler detects a RAM and indicates the depth and width of the RAM. The compiler detects high-level operators such as RAMs, so that the right resources are used to implement these operators during the mapping phase.

0 warnings, 19 notes    Find:     Set Filter...    ☐ Apply Filter    ☒ GroupCommon ID's

Type	ID	Message	Source Location	Log Location	Time	Report
	2	CD720 Setting time resolution to ns	<a href="#">std.vhd (123)</a>	<a href="#">eight_bit_uc.srr</a>	17:05:...	Compili
	10	CG364 Synthesizing module eight_bit_uc	-	<a href="#">eight_bit_uc.srr</a>	17:05:...	Compili
	[2]	CG346 Read full_case directive	<a href="#">alu.v (93)</a>	<a href="#">eight_bit_uc.srr (32)</a>	17:05:...	Compili
		CG794 Using module INS_ROM from library work	<a href="#">eight_bit_uc.v (79)</a>	<a href="#">eight_bit_uc.srr (81)</a>	17:05:...	Compili
	2	CL201 Trying to extract state machine for register state	-	<a href="#">eight_bit_uc.srr</a>	17:05:...	Compili
		CL134 Found RAM mem_regfile, depth=32, width=8	<a href="#">reg_file.v (17)</a>	<a href="#">eight_bit_uc.srr (68)</a>	17:05:...	Compili
		CD630 Synthesizing work.ins_rom.first	<a href="#">ins_rom.vhd (13)</a>	<a href="#">eight_bit_uc.srr (118)</a>	17:05:...	Compili

TCL Script    Messages

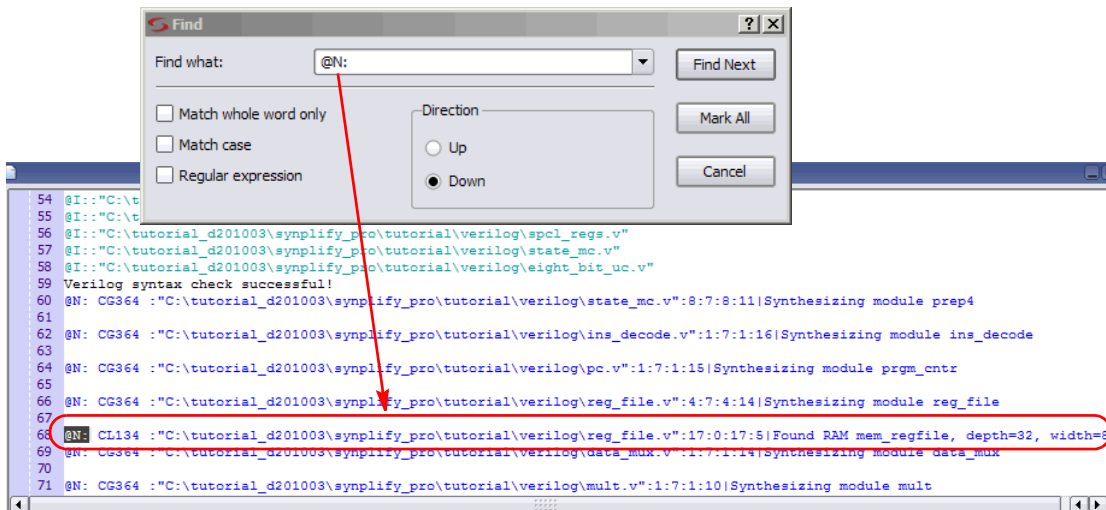
3. If you prefer, review the messages in the Log file. You can perform any of the following tasks:

- Enable the View Log File in HTML option under Options->Project View Options. Click View Log to open the log file and click Compiler Report in the left panel. Scroll through the log file messages on the right.
- Double-click on the log file in the Project view and press Ctrl-f to open the Find dialog box. Enter @N as the search criteria, and click Find Next. The pointer moves to the first note in the log file. Click Find Next again until you find message number (CL134).

Review all messages and then click Cancel in the Find dialog box. If needed, you can also enter the following search criteria: @I for informational messages and @W for warnings.

- Single click the message number (CD134) in the HTML Log file, or double-click the message number in the text-based Log file to open an online help page with information about the message.





4. Review all messages. For this exercise, all messages are valid.

## Additional Analysis after Compile

Once you have a compiled design, you can perform additional analysis before completely synthesizing the design. These include:

- Viewing an RTL schematic of the design
- Crossprobing between the schematic, source code and log file
- Using Object Find in the RTL view
- Filtering and expanding the schematic

See [Appendix A: Early Analysis \(Compile Phase\)](#), on page 50 for details.

# Setup Implementation for Synthesis

This synthesis flow for Xilinx uses the Virtex-6 technology. The following sections discuss these topics:

- [Set Constraints](#), next
- [Set Device Options](#), on page 30
- [Run Logic Synthesis](#), on page 39
- [Analyze Logic Synthesis Results](#), on page 40


If you do not use Xilinx technology, you can follow along with the tutorial using device options specific to your vendor. However, for the following section of this tutorial, make sure that the Xilinx Virtex-6 technology is selected on the Implementation Options dialog box of the Device tab. See [Set Device Options](#), on page 30.

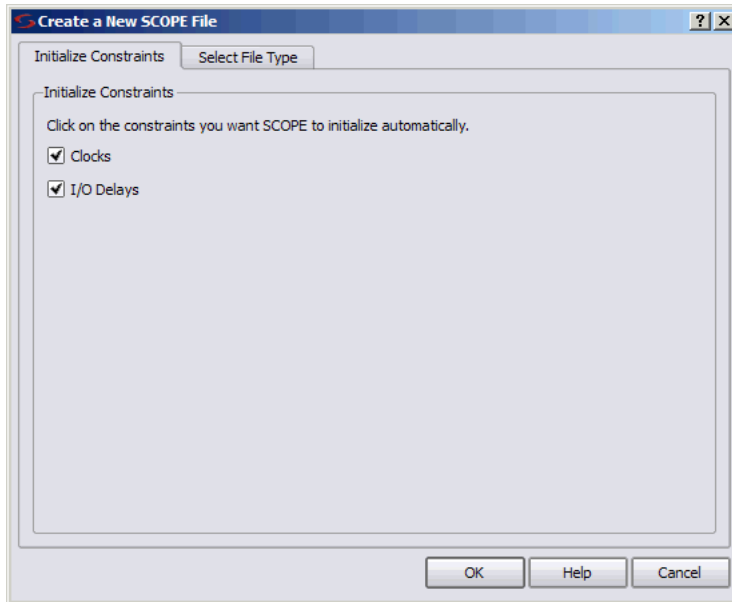
## Set Constraints

Design constraints are optional, but most designers use them to define frequency goals and describe the environment for the design. For designs without aggressive timing goals, you can just set the clock frequency.

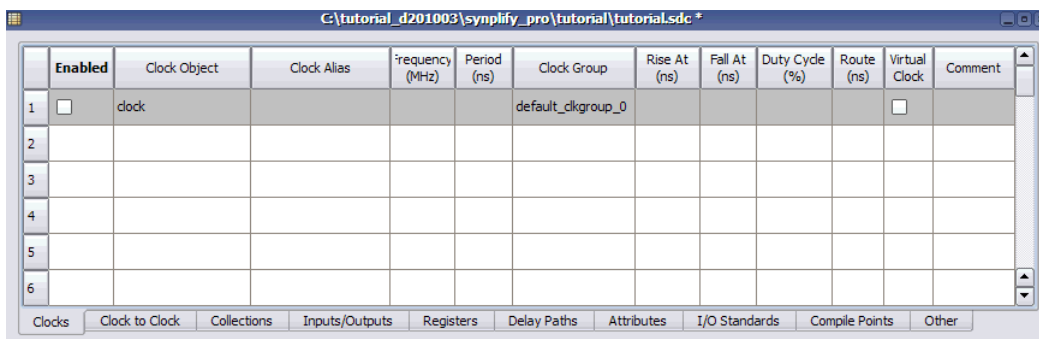
You can set constraints in a text file that you can create with any text editor, but it is easier to use the SCOPE (Synthesis Constraint Optimization Environment) interface. The SCOPE interface consists of a spreadsheet where you enter constraints.

The tutorial design uses basic constraints, which you enter as follows:

1. Start the SCOPE interface in the open project window by doing one of the following:
  - Click the New Constraint file (SCOPE) icon in the toolbar. ()
  - Select File->New, choose Constraint file (SCOPE) in the dialog box, specify the file name (tutorial.sdc) and click OK.
2. Click OK on the Initialize Constraints tab in the Create a New SCOPE File dialog box.



The SCOPE window opens, with the most common constraints, clock frequency and input/output delays initialized. The window consists of a spreadsheet interface with tabs for different kinds of constraints.



### 3. Set a clock frequency constraint as follows:

- Select the Clocks tab at the bottom of the SCOPE window, if it is not already selected.
- Select the check box in the Enabled column to enable the clock constraint.

- Enter 188 in the Frequency column to set the clock frequency and press Enter.

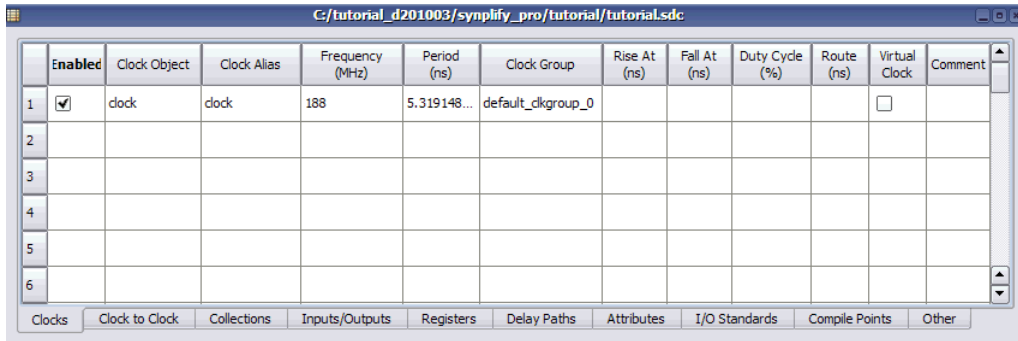
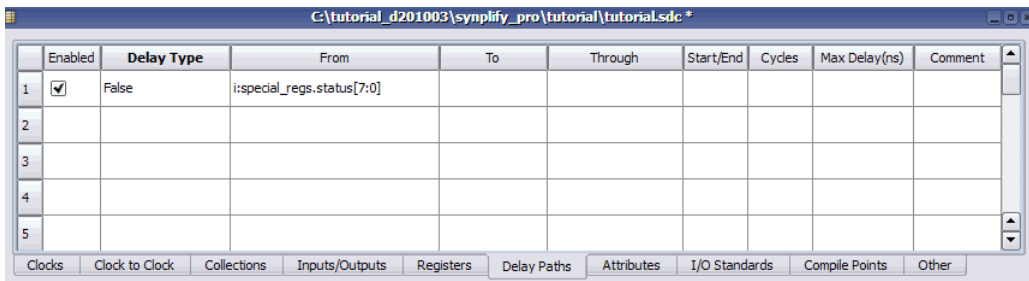


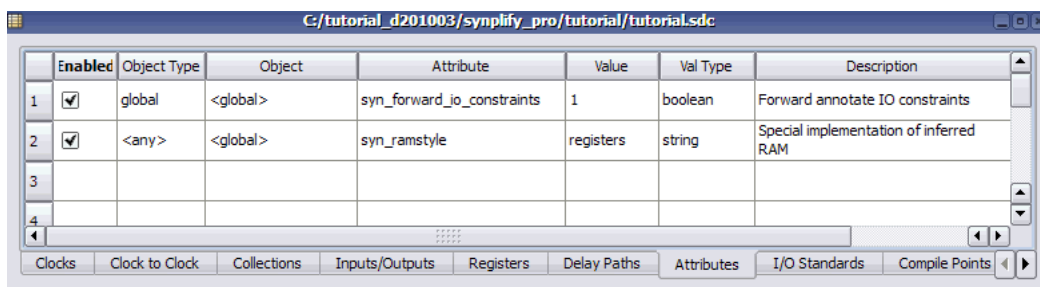
Figure 8: Set Clock Frequency to 188


This design has only one clock, so setting the clock frequency is the same as setting a global frequency from the Project view. When you press Enter, the software automatically sets the clock period and assigns the clock to the default clock group.

- For this exercise, set a false path constraint. Perform the following:
  - Select the Delay Paths tab at the bottom of the SCOPE window.
  - Select the check box in the Enabled column to enable the false path constraint.
  - In the Delay Type field of SCOPE, select False from the drop-down menu.
  - In the From field of SCOPE, select `i:special_regs.status[7:0]` from the drop-down menu.



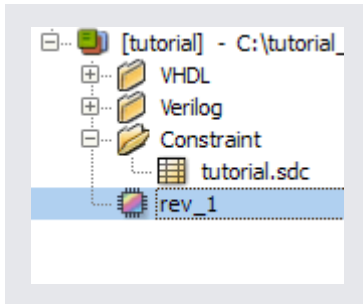
5. Also for this exercise, you will set some attribute constraints. Perform the following tasks:
  - Select the Attributes tab at the bottom of the SCOPE window.
  - Select the check box in the Enabled column to enable the attributes specified below.
  - From the Attributes field, select `syn_forward_io_constraints` from the drop-down menu and press Enter. Leave the default setting for all other fields of this attribute.
  - From the Attributes field, select the `syn_ramstyle` attribute from the drop-down menu. Then, select registers from the drop-down menu in the Value field.



6. Click the Save (  ) icon or select File->Save and save the file as `tutorial.sdc`.
7. Click Yes in the dialog box that asks you if you want to add the file to your project and close the SCOPE window.

You should now have the following files in the project:

- A Verilog folder that contains the source files
  - A VHDL folder that contains the source files
  - A Constraint folder with the constraint file (`tutorial.sdc`)
  - An implementation folder (`rev_1`)
8. Close the SCOPE file.



## Set Device Options

The options you set for a project revision (implementation) determine the optimization settings and inputs such as the device technology, constraint files, and output directory for the synthesis run.

1. Make sure rev\_1 is the current implementation (highlighted). You can bring up the Options for Implementation dialog box in the Project view with one of the following methods:

- Impl Options button
- Select Project->Implementation Options
- New Impl button (for creating a new implementation only)

The Options for Implementation dialog box lists the implementation (rev\_1) at the top.

2. This dialog box has many tabs, and opens with the Device tab displayed. For this exercise:
  - Technology should already be set to Xilinx Virtex6.
  - Use the following technology defaults of: Part XC6VLX75T, Speed -1, and Package FF484.
  - Do not change the default settings for the Device Mapping Options.

Device

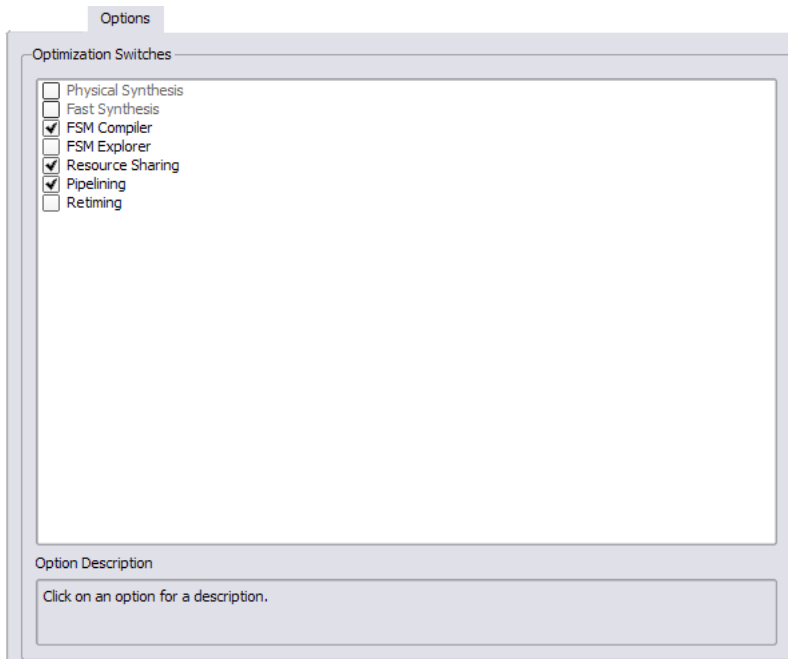
Technology:  Part:  Package:  Speed:

Device Mapping Options

Option	Value
Enhanced Compile Point Support (Beta Support)	<input type="checkbox"/>
Fanout Guide	10000
Disable I/O Insertion	<input type="checkbox"/>
Disable Sequential Optimizations	<input type="checkbox"/>
Fix Gated Clocks	3
Fix Generated Clocks	3
Update Compile Point Timing Data	<input type="checkbox"/>
Use Xilinx Partition Flow	<input type="checkbox"/>
Use Xilinx Xflow	<input type="checkbox"/>
Enable Advanced LUT Combining (use only with Xilinx ISE 10.1 sp1 or later)	<input checked="" type="checkbox"/>
Annotated Properties for Analyst	<input checked="" type="checkbox"/>
Verification Mode	<input type="checkbox"/>

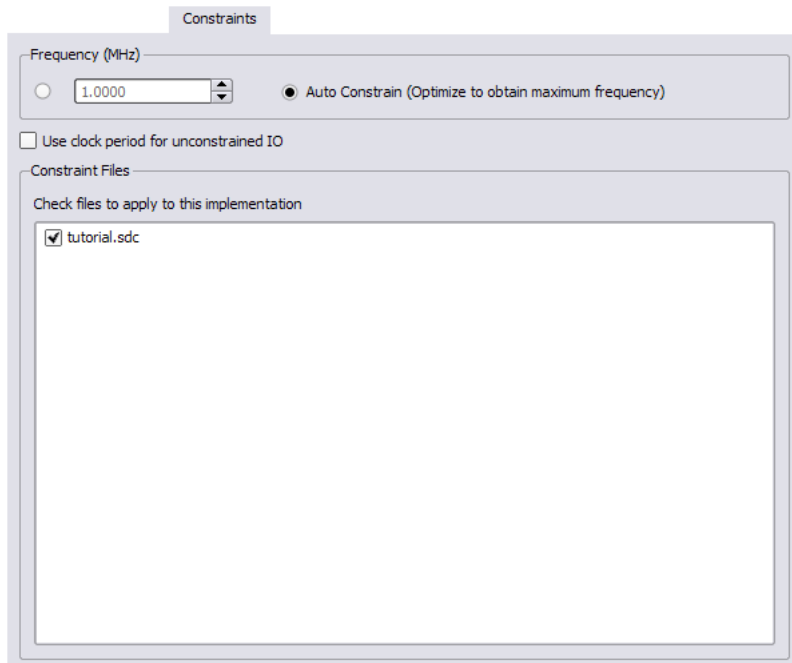
Click on an option for description

3. Click on the Options tab. For this exercise, do not change the default optimization switches for:
- FSM Compiler
  - Resource Sharing
  - Pipelining

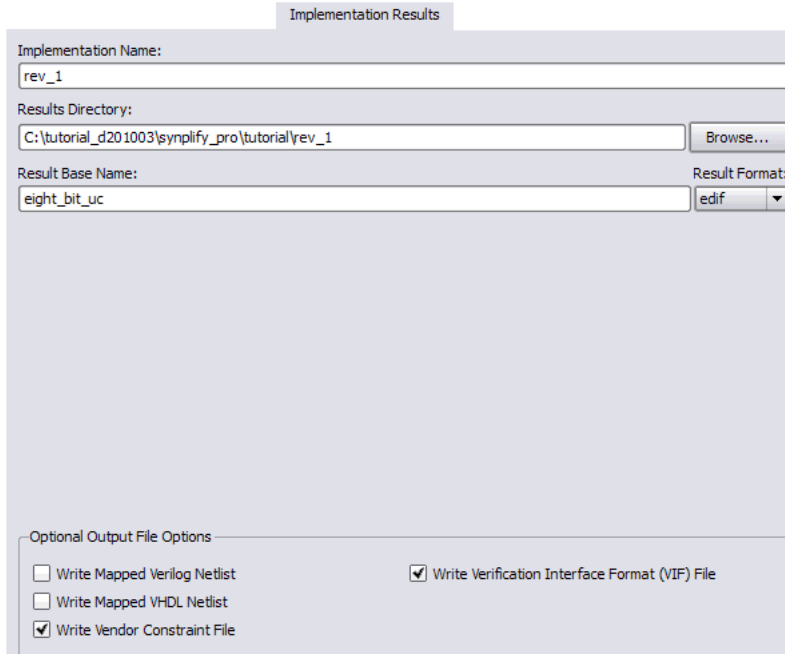


4. Click on the Constraints tab. Make sure the constraint file (tutorial.sdc) is checked.





5. Click on the Implementation Results tab. Make sure that:
- Implementation Name, Results Directory, and Result File Name fields automatically get filled in.
  - Result Format should be edif.
  - Write Vendor Constraint File is enabled.
  - Write Verification Interface Format (VIF) File is enabled.



The image shows a dialog box titled "Implementation Results". It contains several input fields and checkboxes. The "Implementation Name:" field is set to "rev\_1". The "Results Directory:" field is set to "C:\tutorial\_d201003\synplify\_pro\tutorial\rev\_1", with a "Browse..." button to its right. The "Result Base Name:" field is set to "eight\_bit\_uc". The "Result Format:" dropdown menu is set to "edif". At the bottom, there is a section titled "Optional Output File Options" containing three checkboxes: "Write Mapped Verilog Netlist" (unchecked), "Write Mapped VHDL Netlist" (unchecked), and "Write Vendor Constraint File" (checked). There is also a checkbox for "Write Verification Interface Format (VIF) File" which is checked.

Implementation Results

Implementation Name:  
rev\_1

Results Directory:  
C:\tutorial\_d201003\synplify\_pro\tutorial\rev\_1 Browse...

Result Base Name:  
eight\_bit\_uc

Result Format:  
edif

Optional Output File Options

☐ Write Mapped Verilog Netlist ☒ Write Verification Interface Format (VIF) File

☐ Write Mapped VHDL Netlist

☒ Write Vendor Constraint File

6. Click on the Timing Report tab. Set Number of Critical Paths to 25.

This option determines the number of critical paths reported in the timing report generated after synthesis.



Timing Report

Number of Critical Paths: 25

Number of Start/End Points:

Description

Configure the timing report by specifying the number of paths to include in the "Starting/Ending Points with worst slack" and "Worst Paths" report sections.

7. Click on the Verilog tab. For this exercise:

- Check that the Top Level Module is specified as `eight_bit_uc`.
- Leave the default for Verilog 2001 enabled.
- Leave the default for Push Tristates enabled.

Verilog

Top Level Module:  
eight\_bit\_uc

Verilog Language

☒ Verilog 2001  
☐ System Verilog

☒ Push Tristates  
☐ Allow Duplicate Modules  
☐ Multiple File Compilation Unit

Compiler Directives and Parameters

Parameter Name	Value

Extract Parameters

Compiler Directives: e.g. SIZE=8

Include Path Order: (Relative to Project File)

Library Directories:

8. Click on the VHDL tab. For this exercise:

- Select default from the Default Enum Encoding drop-down menu for the top-level entity (`eight_bit_uc`).
- Leave the default setting for Push Tristates enabled.

VHDL

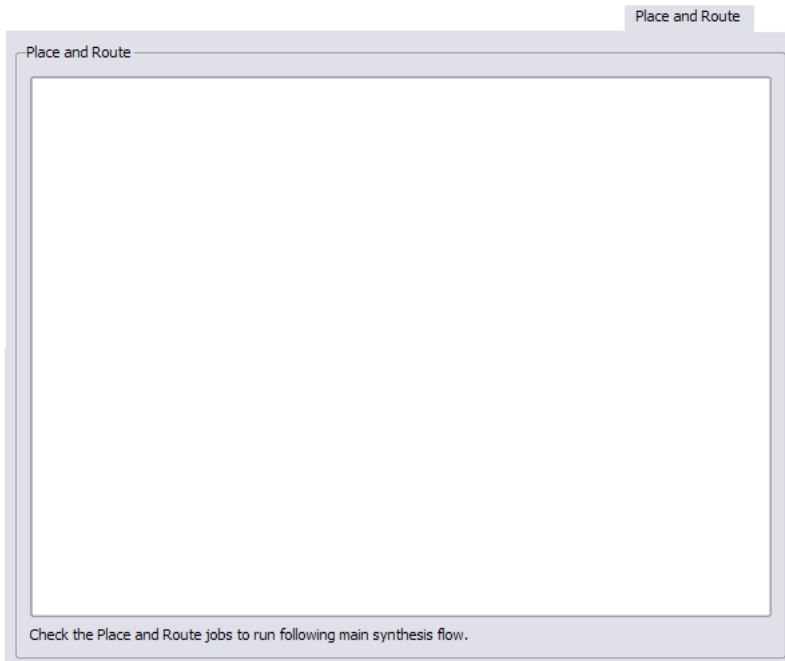
Top Level Entity:  Default Enum Encoding:

☒ Push Tristates  
☐ Synthesis On/Off Implemented as Translate On/Off  
☐ VHDL 2008 (Beta)

Generics

Generic Name	Value

9. Click on the Place and Route tab. You can ignore this option for this exercise, since placement and routing will not be run.



10. Click OK to save the implementation options for rev\_1.

The exercise in this section presents a summary of the implementation options as they relate to the tutorial exercise. For more details on setting implementation options, see the *Setting Up a Logic Synthesis Project* chapter of the *Synopsys FPGA Synthesis User Guide*.

# Run Logic Synthesis

After you have set up your project with source files, implementation settings, and an optional constraint file, synthesize your design. To do this:

1. Click the Run button to start synthesis.

The software goes through two synthesis phases, compilation and mapping, and it reflects these stages in large red letters in the status area.

- Compilation is the creation of a technology-independent boolean structure, and mapping is the technology-specific implementation and optimization of the boolean structure.
- You can see the results of compilation in the RTL view. Mapping results are displayed in the Technology view, which is described in more detail in subsequent sections.

2. When synthesis is complete, the software usually displays this message:



Done:

Note that the Implementation Results view lists the files that are generated as a result of synthesis.


# Analyze Logic Synthesis Results

After you have run synthesis, you can analyze the results. This section shows you how to do the following:

- [Examine the Technology View](#), next
- [Check Timing](#), on page 42
- [Analyze Critical Paths in the Technology View](#), on page 45

## Examine the Technology View

You can graphically check the synthesis results in the Technology view.

1. To see the graphical results of your run, click the  icon on the menu bar to open the Technology view.

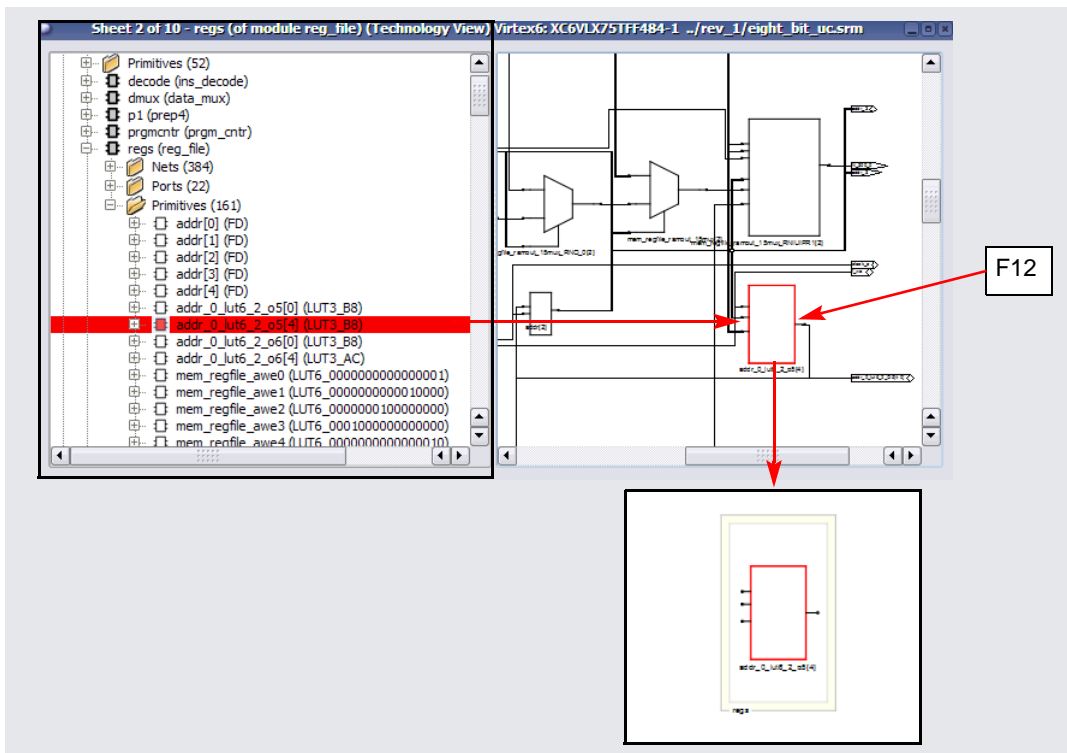
The Technology view contains a schematic of the design after technology mapping with base cells that are directly mapped to the target technology.

2. Examine one of the technology-specific components as described below. If you are not using the version of software for which this tutorial was written, your design may be implemented with different components because of ongoing optimizations to the technology and the software.
  - To reduce congestion in the schematic, select Options->HDL Analyst Options and disable Show cell interior on the General tab if it is on. You can also disable the display of symbol and pin names on the Text tab. Click OK.
  - In the Hierarchy Browser on the left side of the Technology view, expand Instances and then expand one of the modules. In the following example, we chose REGS (REGS\_FILE).

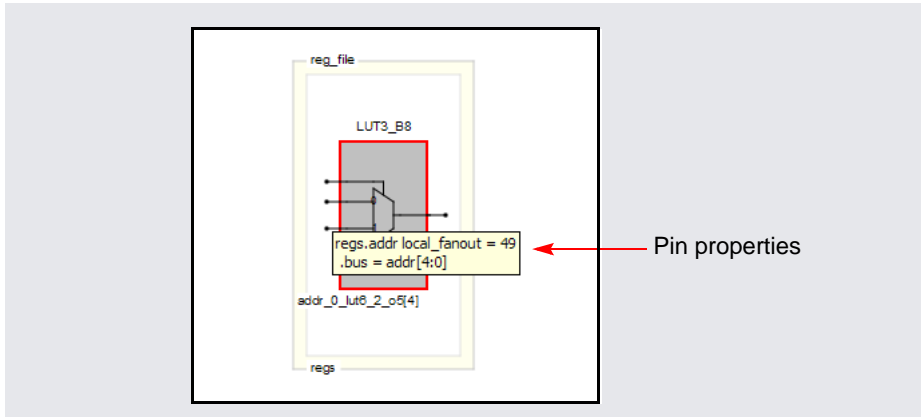


- Next expand Primitives and select a primitive instance. The instance selected is highlighted in red on the schematic. When you have multiple sheets, the Technology view automatically moves to the sheet with the selected component.

**Note:** Small sheet size is a preference; you can set with Options->HDL Analyst Options->Sheet Size.



- Filter the selected component. To filter, click F12, the Filter icon, or click the right mouse button and select Filter Schematic. You see just the object selected.
- To see details of this object, select Options->HDL Analyst Options and enable Show symbol name on the Text tab, and Show cell interior on the General tab. Click OK. You see the interior of the cell. You can see any properties attached to the pins, like fanout.



- Deselect the component by clicking in an empty area of the schematic.
- Use the techniques described in [Additional Analysis after Compile, on page 25](#), [Find and Crossprobe, on page 55](#), and [Filter, Expand, Hide, and Dissolve, on page 61](#) to examine how the design was implemented for this technology.
- When finished, close the Technology view window.

## Check Timing

You can check timing results in the log file and in the Log Watch window.

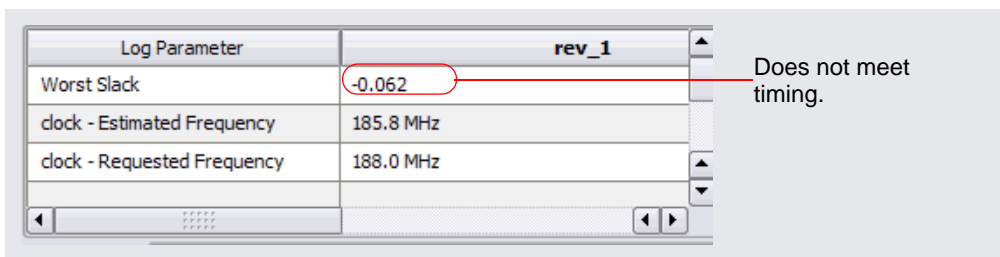
### Using the Log Watch Window

In the tool, the Log Watch Window is a quick way to view just the critical timing.

1. Check the timing parameters in the Log Watch window:
  - If you do not already have it open, select View -> Log Watch Window to open a window where you can quickly see the critical timing information in a tabular format.
  - Position the cursor in the first cell in the Log Parameter column, and hold down the left mouse button.

- Select Worst Slack from the drop-down list. The software displays the corresponding value.
- Use the same method to set clock - Estimated Frequency and clock - Requested Frequency in subsequent cells. You can see that the design does not meet timing because it has a negative slack value. Positive or 0 slack times indicate that you have met or are within the timing constraint. Close the Log Watch window.

The following figure shows the values in the Log Watch window after the run. If you are using a different release of the software, the values you get when you run the tutorial might vary slightly, because of ongoing optimizations within the synthesis tool.



Log Parameter	rev_1
Worst Slack	-0.062
clock - Estimated Frequency	185.8 MHz
clock - Requested Frequency	188.0 MHz

## Using the Log File

The log file is available in text format, as well as, the HTML-based viewer for the Synplify Pro tool. To enable the HTML version of the log file, select Options->Project View Options->View log file in HTML.

To see detailed information about the critical paths, open the log file (eight\_bit\_uc.srr) by clicking the View Log button. You see a window with the log file.

- In the text-based log file window, scroll down to the Performance Summary section to see details of the clock information. Scroll a little further to the Worst Paths Information section. (You can also use Ctrl-f and search for Worst Paths.) A table shows all the points on the critical path.

```

423
424 Path information for path number 1:
425     Requested Period:                5.319
426     - Setup time:                    -0.045
427     + Clock delay at ending point:    3.004
428     = Required time:                 8.368
429
430     - Propagation time:               5.427
431     - Clock delay at starting point:  3.004
432     = Slack (critical) :              -0.062
433
434     Number of logic level(s):        23
435     Starting point:                  dmux.alubtmp_fast[0] / Q
436     Ending point:                    uc_alu.aluz / D
437     The start point is clocked by    clock [rising] on pin C
438     The end point is clocked by      clock [rising] on pin C
439

```

- In the HTML log file window, select Worst Path Information in the left table of contents pane of the window.

```

rev_1 (eight_bit_uc)
Compiler Report
Mapper Report
Timing Report
Performance Summary
Clock Relationships
Interface Information
Detailed Report for Clock: clock
Starting Points with Worst Slack
Ending Points with Worst Slack
Worst Path Information
Resource Utilization

Log File Links:
rev_1
Hierarchical Area Report (eight_bit_uc)

Worst Path Information
View Worst Path in Analyst
*****

Path information for path number 1:
    Requested Period:                5.319
    - Setup time:                    -0.045
    + Clock delay at ending point:    3.004
    = Required time:                 8.368

    - Propagation time:               5.427
    - Clock delay at starting point:  3.004
    = Slack (critical) :              -0.062



    Number of logic level(s):        23
    Starting point:                  dmux.alubtmp_fast[0] / Q
    Ending point:                    uc_alu.aluz / D
    The start point is clocked by    clock [rising] on pin C
    The end point is clocked by      clock [rising] on pin C

```

The worst path doesn't meet timing as indicated by the negative slack value. You can now check the critical path in the Technology view.

## Analyze Critical Paths in the Technology View

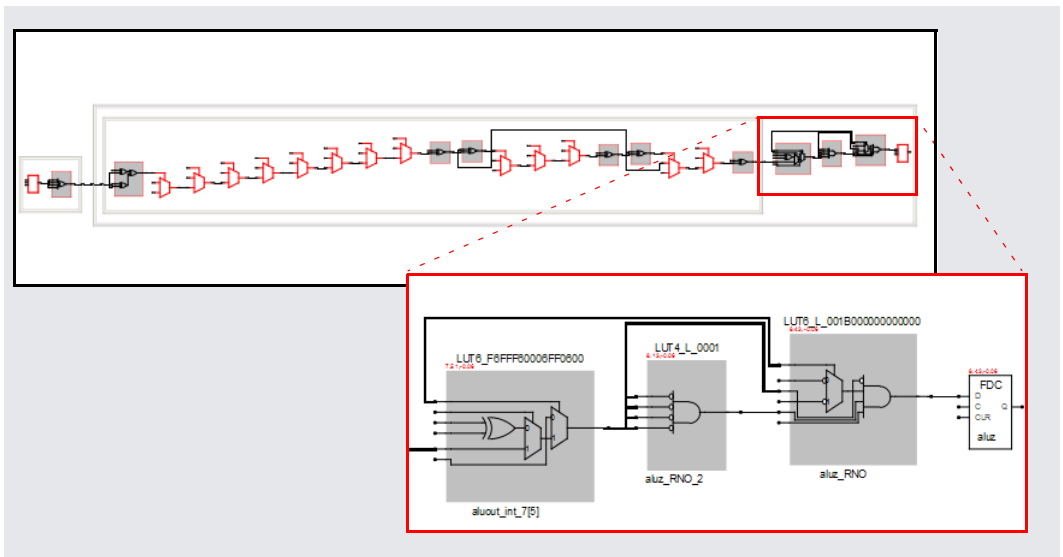
To analyze your critical path in the Technology view, do one of the following:

1. Open a Technology view window by clicking the and gate icon  in the menu bar.
2. Select the Critical Path icon  on the menu bar or right-click in the Technology view window and select Show Critical Path.
3. In the HTML log window, select the View Worst Path in Analyst link at the beginning of the Worst Path Information section.

The Technology view graphically displays the path described in the log file. The critical path view is a *filtered* view that shows only the instances on the critical path.

The following figure shows the critical path with transparent instances to indicate the design hierarchy. To display the cell interiors, select Options->HDL Analyst Options->General->Show cell interior.

You should see red numbers at the upper left corners of the instances. These numbers provide timing information: the first value is the cumulative delay, and the second value is the total slack time for the path. If the red timing information is not displayed, enable HDL Analyst->Show Timing Information.



4. Zoom in to the timing information. You can see that the slack (second number) is negative, which means that your design does not meet timing.
5. You can now use other techniques to analyze your path and design further. For example:
  - Check the corresponding RTL code by double-clicking objects in the Technology view.
  - Filter and expand paths using the techniques described in [Filter, Expand, Hide, and Dissolve, on page 61](#).
  - To return to the critical path view, click Back or click the Critical Path icon. If Back is inactive (the path has been flattened), click the Critical Path icon to return to the critical path view.

For this tutorial, you will reduce the delay on this critical path by adding a two-cycle path constraint and resynthesizing the design. See the *Synopsys FPGA Synthesis Reference Manual* for details about other constraints and attributes you can add.

6. Leave the filtered critical path view open, and close any other open Technology views.

## Improve Results

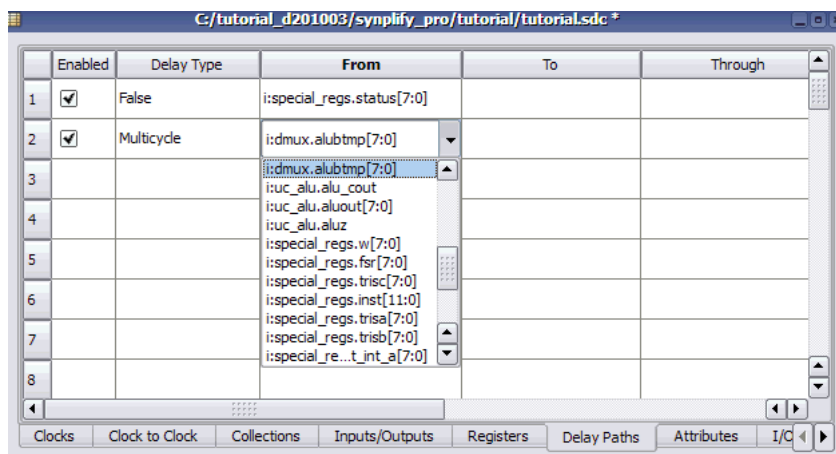
This section guides you through the post-analysis phase, where you fine tune your design by setting constraints, rerunning synthesis, and checking your results.

### Set Additional Constraint and Resynthesize

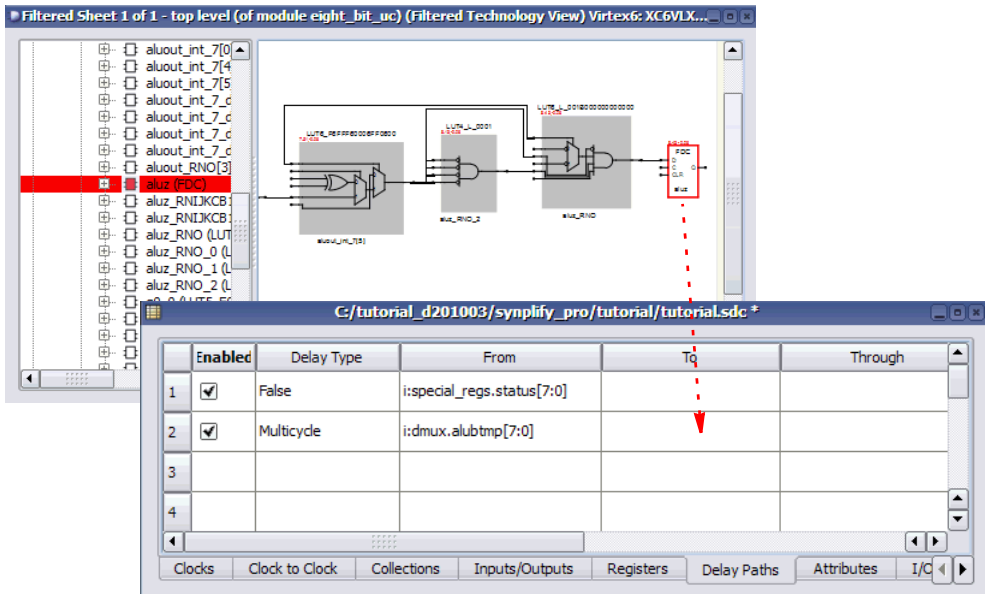
Since the design did not meet timing, you can add a timing constraint to the critical path in the constraint file, then resynthesize the design.

1. Make sure you have the filtered view of the critical path open.
2. Open the constraints file (tutorial.sdc) and select the Delay Paths tab.

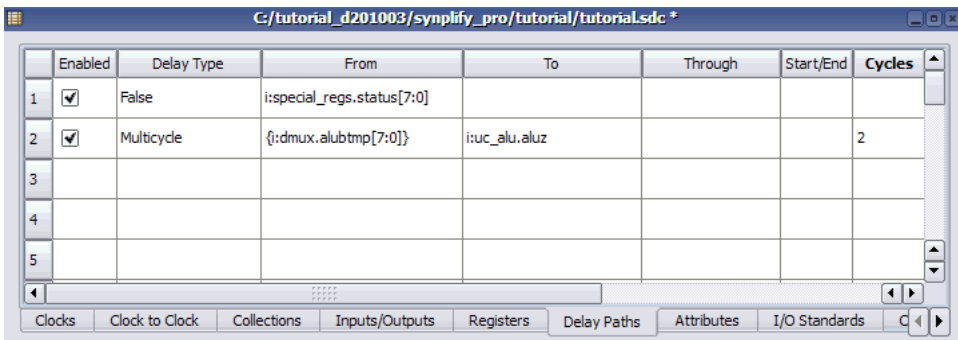
3. Select the check box in the Enabled column to enable the false path constraint.
4. In the Delay Type field of SCOPE, select Multicycle from the drop-down menu.
5. Add a constraint from the start (From) point to the end (To) point using these steps:
  - Select the i:dmux.alubtmp[7:0] bus from the drop down menu in the From column. This bus includes the first instance (dmux.alubtmp\_fast[0]) in the most critical path. Adding the constraint to the entire bus eliminates the negative slack times in the remaining bus signals.



- With the critical path view open, select the ending point (i:uc\_alu.aluz) and drag it to the To column.



- Set Cycles to 2 and make sure the Enabled column is selected to apply the constraint.



6. Save the constraint file and minimize or close the SCOPE window.
7. Click the Run button to rerun synthesis. You can now check your results to see if you eliminated the negative slack on the path.



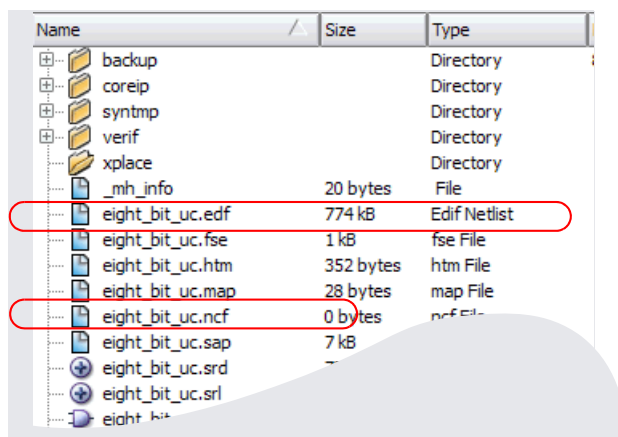
## Check Results

Check the results of the second synthesis run to make sure you achieved your timing goals.

1. Check the results in the Log Watch window or the log file as described previously in [Check Timing, on page 42](#).

The first critical path (and several additional paths associated with the bus) now meets the timing requirements. You see the next most critical path listed as the most critical path. In a design, you would continue to refine your design using constraints, attributes, and other optimizations until you eliminate all the negative slack. For the tutorial, the next most critical path is positive and synthesis is now complete.

2. Check the output files in the Implementation Results view.



Name	Size	Type
backup		Directory
coreip		Directory
syntmp		Directory
verif		Directory
xplace		Directory
_mh_info	20 bytes	File
eight_bit_uc.edf	774 kB	Edif Netlist
eight_bit_uc.fse	1 kB	fse File
eight_bit_uc.htm	352 bytes	htm File
eight_bit_uc.map	28 bytes	map File
eight_bit_uc.ncf	0 bytes	ncf File
eight_bit_uc.sap	7 kB	
eight_bit_uc.srd		
eight_bit_uc.srl		
eight_bit...		

The software generates vendor-specific netlists with the attributes and constraints carried forward to ensure that the design is optimized for the target technology. The `eight_bit_uc.edf` file is the netlist for the place-and-route tools, and the `eight_bit_uc.ncf` file contains the constraints to be passed to the place-and-route tools.

At this point, you have finished synthesis. The next step is to simulate waveforms or to place and route your design. You can use the Synplify Pro interface to crossprobe and debug your designs further, or use the synthesis output files to place and route your design.

# Appendix A: Early Analysis (Compile Phase)


This appendix describes the types of analysis that you can perform after you have compiled your design, and before you click the Run button. After you have created a project and added the project files, including source, and constraint files, you can compile the design (Run->Compile Only, or F7). During the compile phase the RTL view of the design is created and you can use the HDL Analyst features to view the schematic, traverse hierarchy, crossprobe between the view and source code, find design objects and filter and/or expand the logic in the schematic views. Topics in this section include:

- [Analyze Compile Results \(RTL\) and Navigate Hierarchy](#)
- [Find and Crossprobe](#)
- [Filter, Expand, Hide, and Dissolve](#)

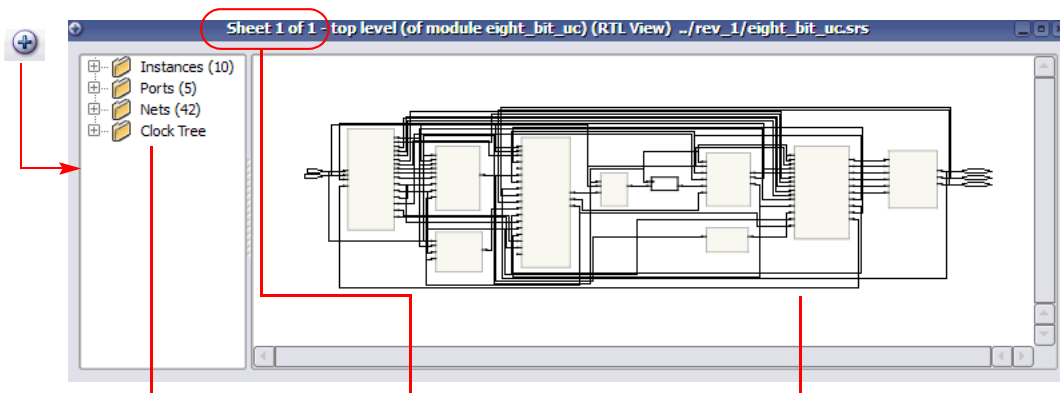
## Analyze Compile Results (RTL) and Navigate Hierarchy

This section covers basic zooming and hierarchy navigation; [Find and Crossprobe, on page 55](#) and [Filter, Expand, Hide, and Dissolve, on page 61](#), with some discussion of other analysis techniques. Synopsys's proprietary BEST (Behavioral Extraction Synthesis Technology) algorithms detect and extract some high-level behavioral constructs in the RTL view. This is different from other synthesis tools, which decompose the RTL into low-level boolean primitives that must be reconstructed into higher-level primitives at the place-and-route stage.

To use the HDL Analyst:

1. Click the RTL View icon from the toolbar (  ) or select HDL Analyst -> RTL -> Hierarchical View to open the RTL view.

To make your view look exactly like the one shown in the following figure, select Options->HDL Analyst Options and on the Text tab, disable the Show pin name option. If your design has more annotations, some of the preferences (Options->HDL Analyst Options) are set differently.








Hierarchy Browser.  
Lists the instances, nets,  
and ports in the design

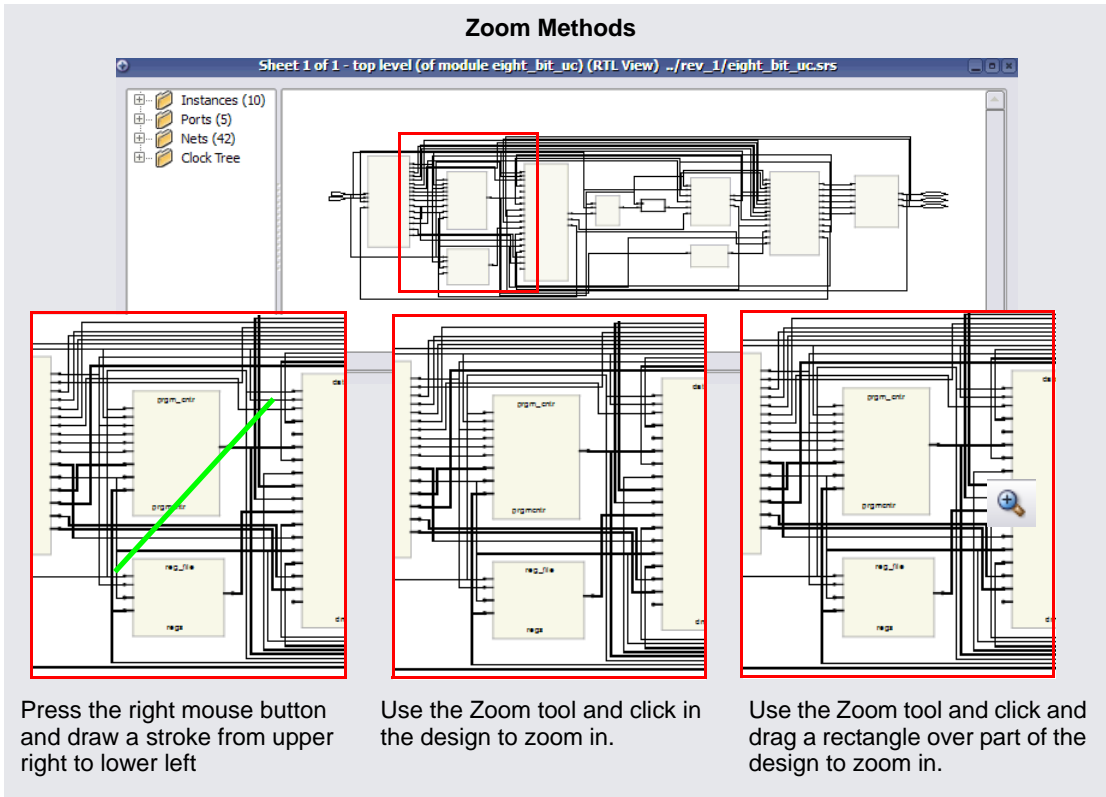
Number of sheets  
in the schematic


Technology-independent  
schematic view

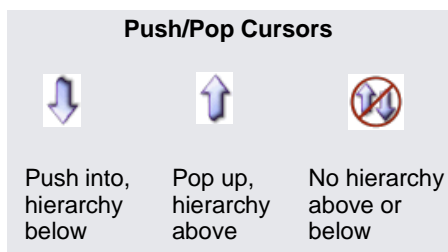
The RTL view is a hierarchical, technology-independent schematic view that is generated by the software. The pale yellow blocks indicate hierarchical instances. The software extracts high-level behavior, represents it as an abstract, and operates on this abstract. You can recognize the high-level blocks of logic from the source code.

2. To view the design, use the sizing icons (     ) from the toolbar, the mouse strokes (see Help->Mouse Stroke Tutor), or the corresponding commands from the View menu.
  - Zoom into the area shown in the following figure by clicking the Zoom In icon (  ) over the area you want to zoom. Click as many times as needed to get a magnification level that is comfortable. You can also zoom by clicking and dragging the icon diagonally to specify a rectangular area for zooming, or by pressing the right mouse button and drawing a diagonal mouse stroke from upper right to lower left over the area to be zoomed. See Help->Mouse Stroke Tutor for a complete list of mouse strokes.

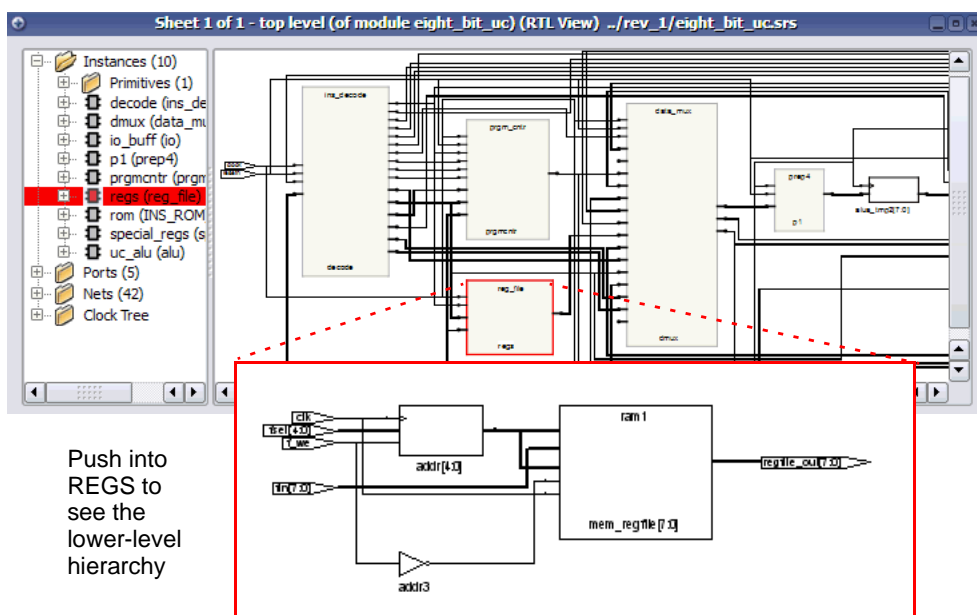
### Zoom Methods






- Exit the zoom mode by clicking on the Zoom in icon again, or by right-clicking in a blank area of the design. The zoom icon changes back to the default crosshair selection cursor.
3. Select the Push/Pop Hierarchy icon (  ) or press F2. The cursor changes to a double-headed arrow with a not sign through it when it is over areas of the design without underlying hierarchy. When it is over a component that has hierarchy below it, the cursor changes to an arrow pointing downward.

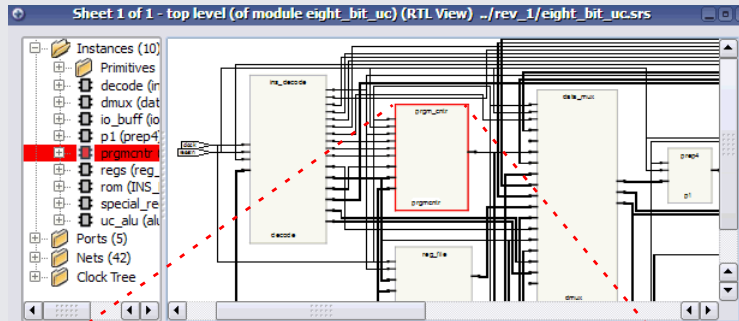


- Click on the REGS block to push down into it. See the lower-level hierarchy and how the software infers the RAM.

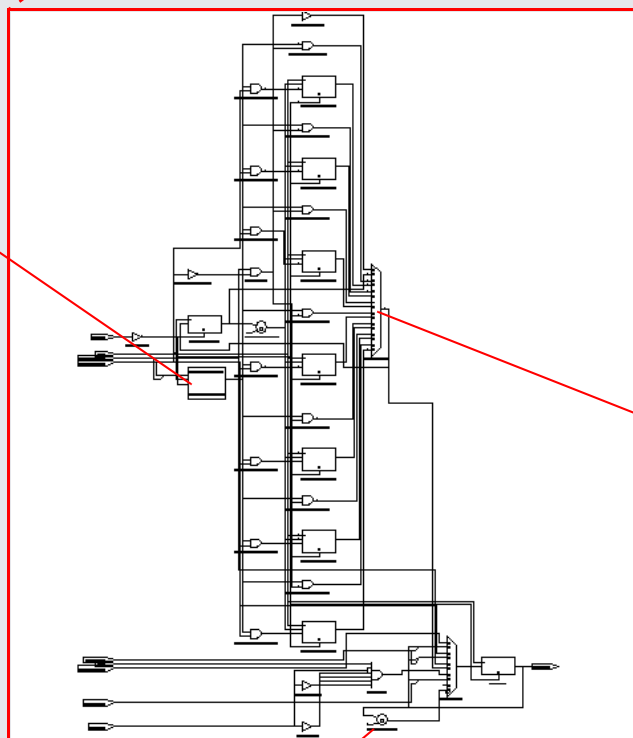


- Pop up to the top level by clicking the up arrow cursor (  ) in an empty area. Alternatively, press the right mouse button and draw a vertical line going upwards in a blank area of the design.
- Page back to the previous view by clicking the Back icon (  ) on the toolbar. Return to the top-level view by clicking the Fwd icon (  ).
- Push down into the Prgm\_Cntr block. To push down with a mouse stroke, press the right mouse button and draw a vertical line going downwards within the block. In the lower-level schematic view shown in the following figure, note the abstracts used to represent high-level behavior: incrementor, state machine, and large mux.

Push into  
PRGM\_CNTR  
to see the  
lower-level  
hierarchy



State machine



Large mux

Incrementor

- Return to the top level and, if necessary, right-click to exit Push/Pop mode.

## Find and Crossprobe

This section shows you how to find objects and crossprobe. For information about other HDL Analyst operations, see [Additional Analysis after Compile, on page 25](#) and [Filter, Expand, Hide, and Dissolve, on page 61](#).

1. With the top-level RTL view open, type Ctrl-f or select Edit->Find.

The Object Query dialog box opens. This dialog box is different from the one that opens when you type Ctrl-f in the Text Editor window.

2. Click the Symbols tab and set the search range to Entire Design.
3. Scroll down in the Unhighlighted box to find the add symbol. Double-click on add to move it to the Highlighted box on the right and click Close.

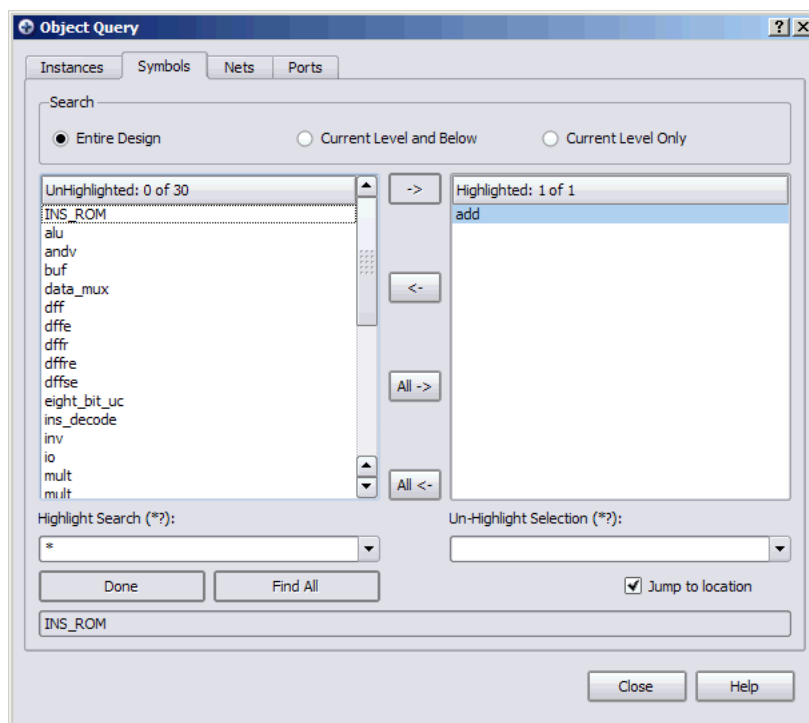


Figure 9: Move add into the Highlighted Field

The software searches the entire design (all hierarchical levels) for the add symbol. The schematic window changes to display lower-level hierarchy (Prgm\_Cntr), with the incrementor (+) highlighted.

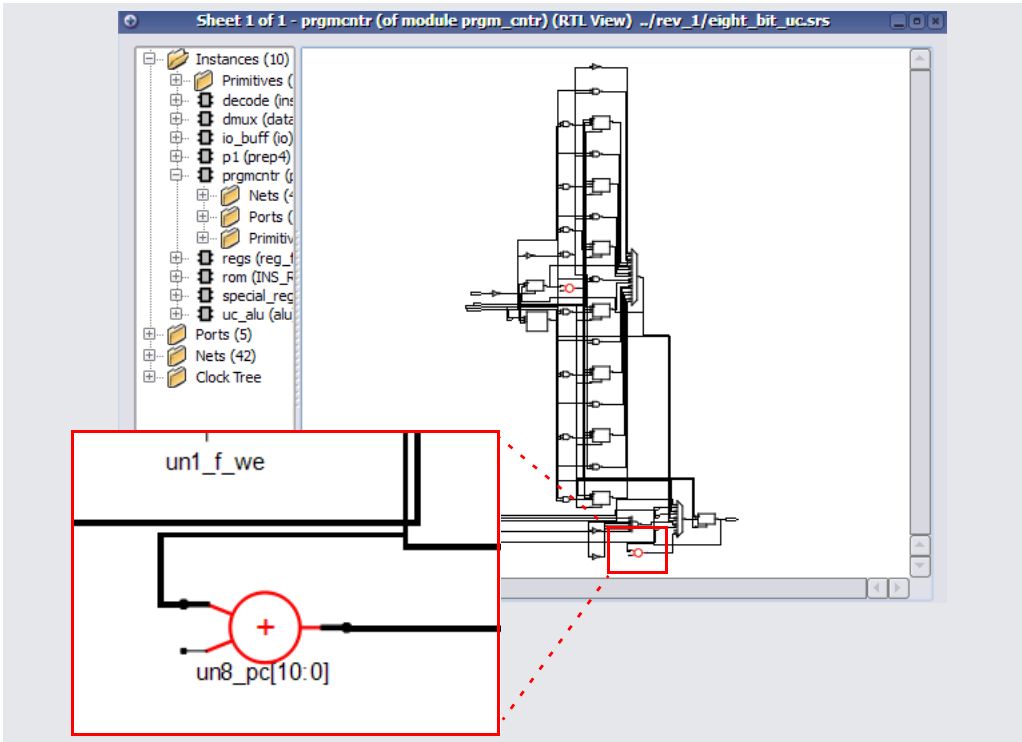


Figure 10: Result of Highlighted add Symbol in the RTL View

4. Crossprobe from the schematic to see the corresponding source code.
  - Double-click on one of the incrementor symbols. The software displays the corresponding RTL code. For example, the following figure shows Verilog source code.



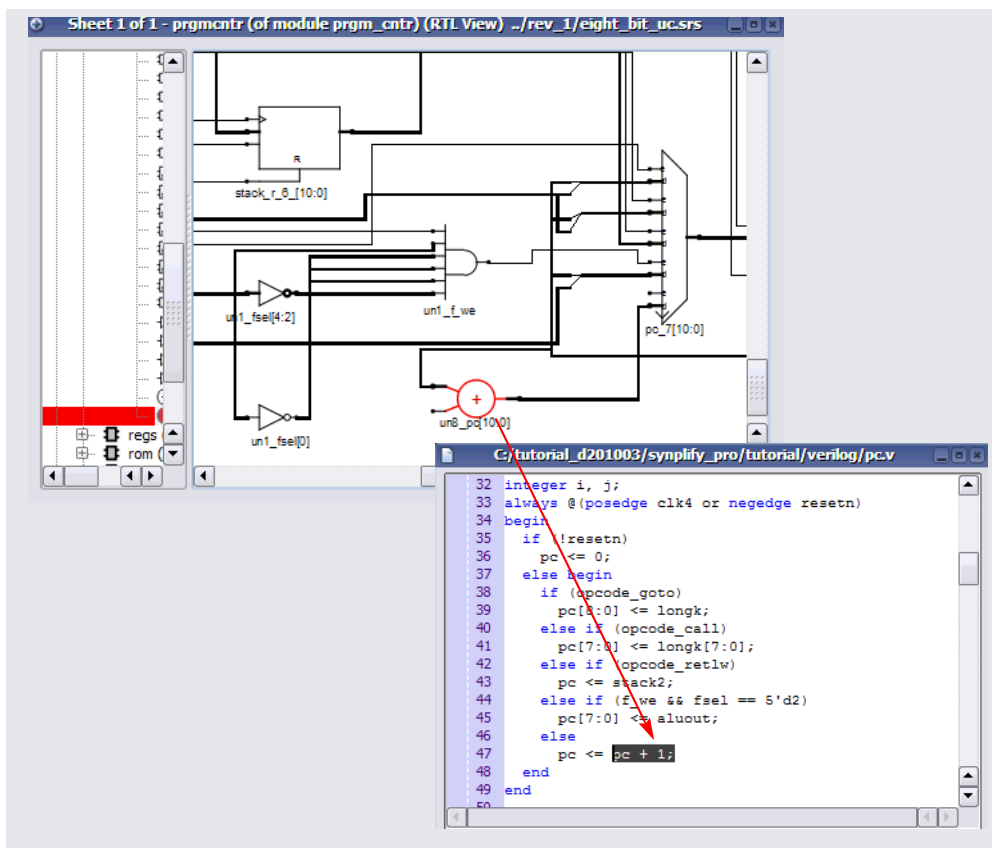


Figure 11: Double-click Incrementor Symbol to Open Source File

- Close the source file window and use Push/Pop mode to return to the top level.
- 5. Crossprobe from the source code to see the corresponding schematic view.
  - From the top-level RTL view, push down into the ALU block.
  - In the Project view, double-click alu.v to open the source code file.
  - In the alu.v file, go to line 92 by pressing Ctrl-g and typing 92.
  - Highlight the section from line 92 that begins with:

```

begin
  case (aluop[2:0])// synthesis full_case

```

to line 104 that ends with:

```
    endcase  
end
```

- Then right-click and select Filter in Analyst from the popup menu.

The corresponding logic is highlighted in the RTL view.

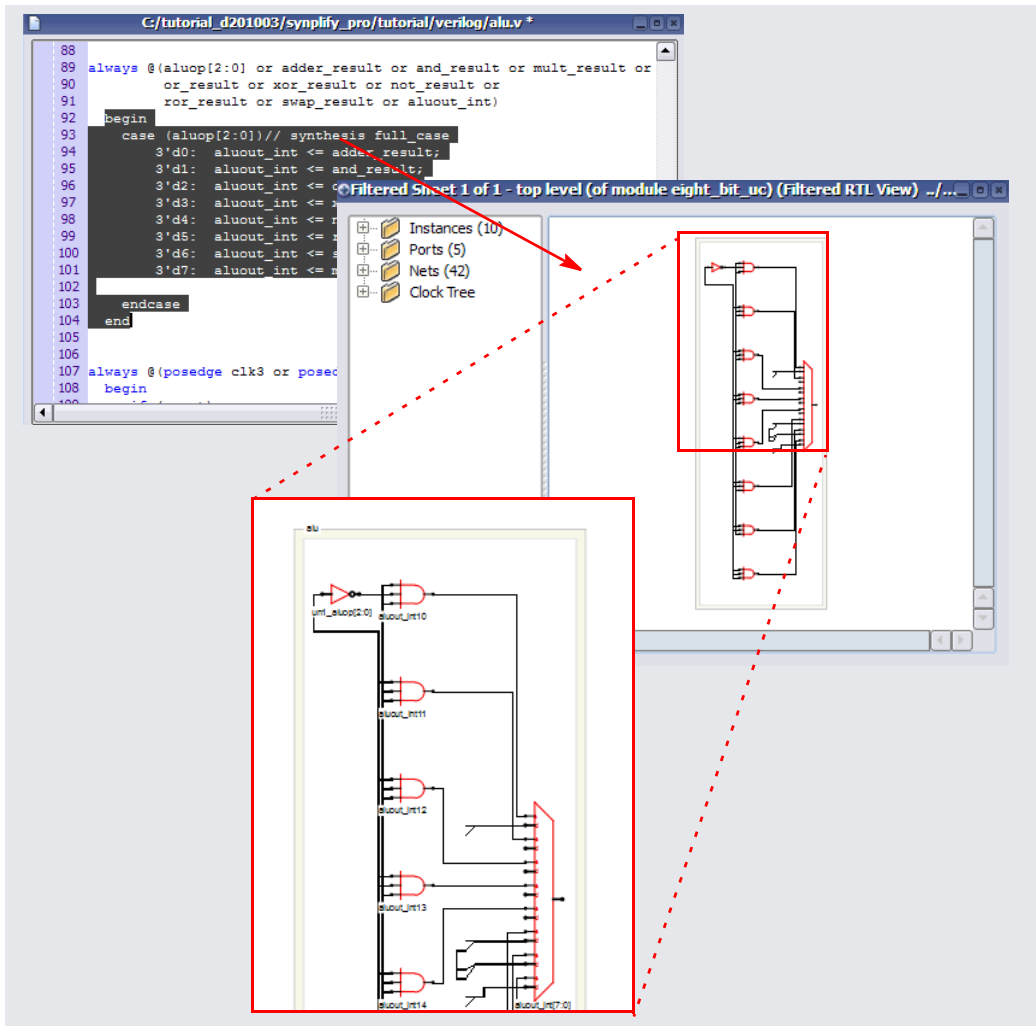
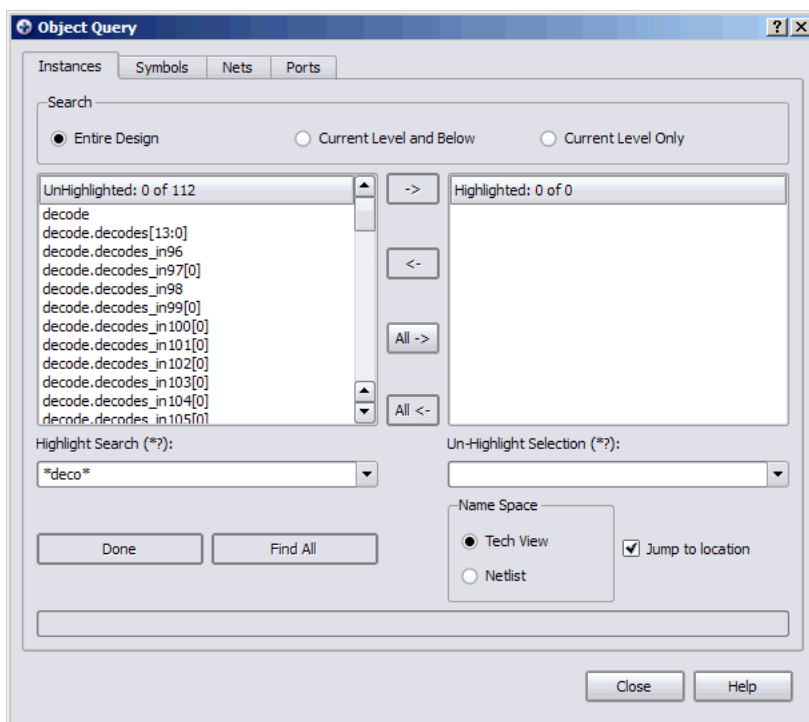


Figure 12: Selecting HDL Code Highlights the Logic in the RTL View

6. Find a bit decoder and crossprobe.
  - Use Push/Pop mode to return to the top level.
  - From the RTL view, press Ctrl-f or select Edit->Find to open the Object Query dialog box.
  - Select the Instances tab and set the search range to Entire Design.
  - In the Highlight Search (\*) field, type `*deco*`, and click Find 200 to find the first 200 occurrences of this string.



The Unhighlighted selection list is now shorter, and only lists instances that match the search criteria. For details about using wildcards, see the *Analyzing with HDL Analyst and FSM Viewer* chapter of the *Synopsys FPGA Synthesis User Guide*.

- Click All-> to move the entire list to the Highlighted box and click Close to close the Object Query dialog box. The schematic highlights the bit decoder instances in red.

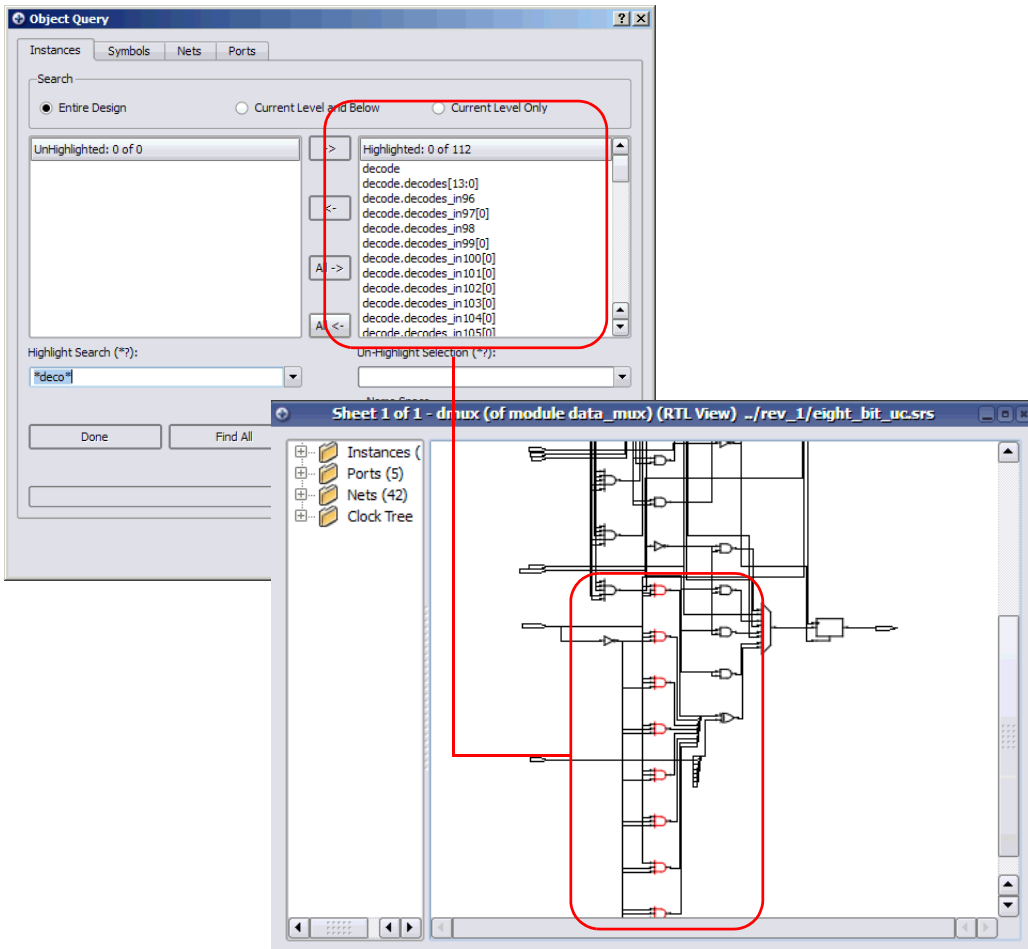
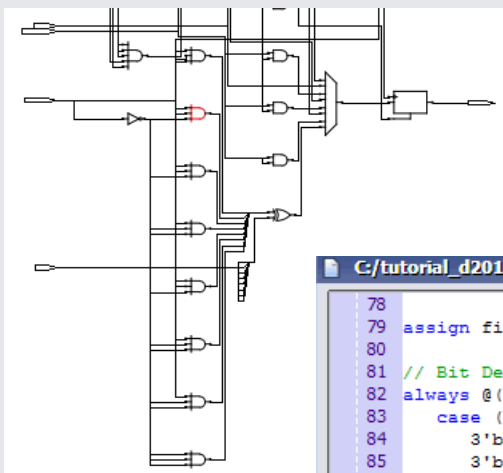


Figure 13: Highlighted List Appears in the RTL View in Red

- With the bit decoder instances selected in the RTL view, put your cursor over one of the selected instances and double-click. The corresponding source code for the bit-decoder definition in the `data_mux.v` file opens.

Double-click on one of these highlighted components in the RTL view...



...to see the corresponding source code.

```

C:/tutorial_d201003/synplify_pro/tutorial/verilog/data_mux.v
78
79 assign fin = aluout;
80
81 // Bit Decoder
82 always @(b_mux) begin
83     case (b_mux)
84         3'b000: bit_decoder <= 8'b00000001;
85         3'b001: bit_decoder <= 8'b00000010;
86         3'b010: bit_decoder <= 8'b00000100;
87         3'b011: bit_decoder <= 8'b00001000;
88         3'b100: bit_decoder <= 8'b00010000;
89         3'b101: bit_decoder <= 8'b00100000;
90         3'b110: bit_decoder <= 8'b01000000;
91         3'b111: bit_decoder <= 8'b10000000;
92     endcase
93 end
94
95


```

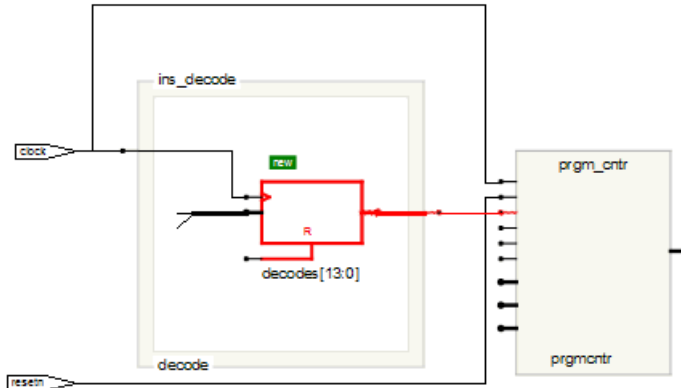
To crossprobe from the bit-decoder definition to the schematic, the RTL window must be open. You can select any number of bits that make up the bit-decoder definition in the source code.

7. Close the source code window and return to the top-level schematic view.

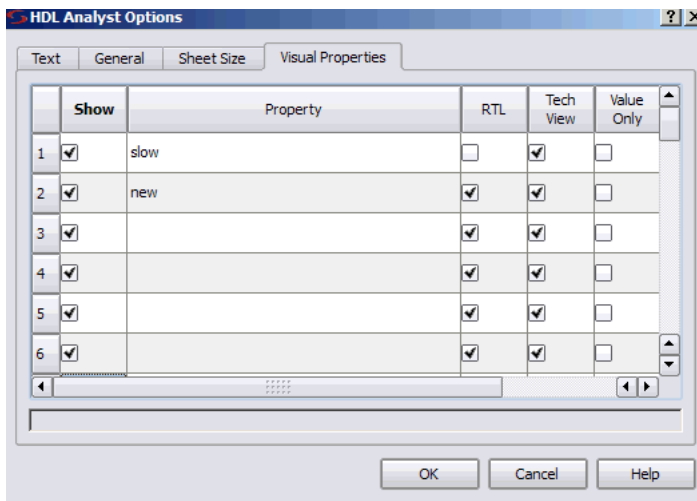
## Filter, Expand, Hide, and Dissolve

Now that you are familiar with basic zooming and push/pop navigation (see [Additional Analysis after Compile, on page 25](#)), you can filter, expand, and dissolve parts of your design for analysis. This is a quick overview; for a more detailed discussion, refer to the *Analyzing with HDL Analyst and FSM Viewer* chapter of the *Synopsys FPGA Synthesis User Guide*.

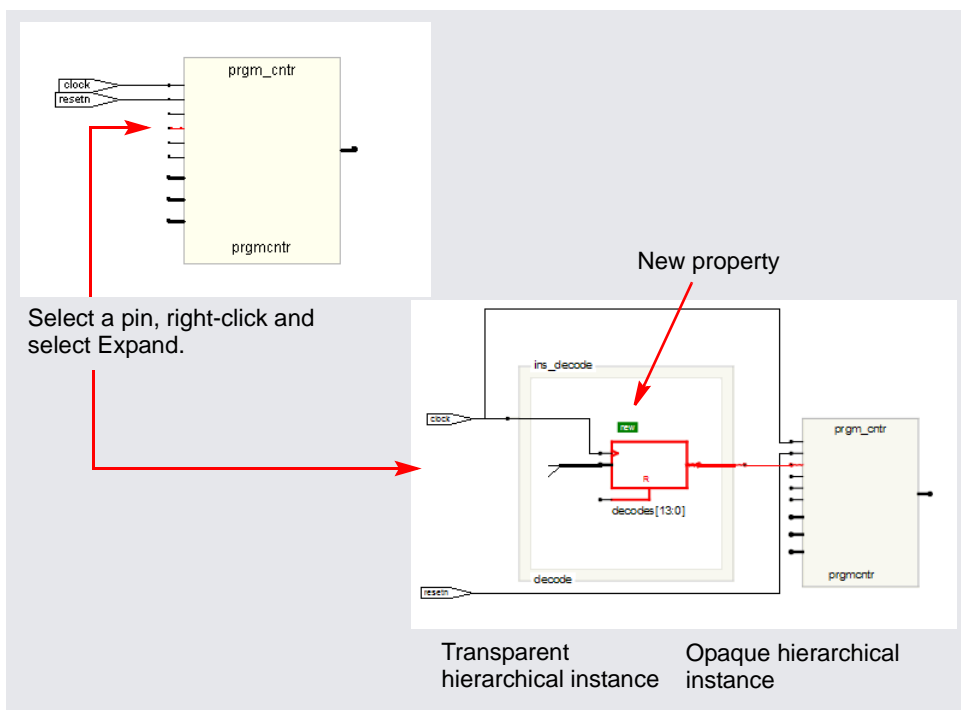
1. In the top-level RTL view, filter and expand pin connections:
  - Select Prgm\_Cntr and press F12 or select the Filter on Selected Gates icon (  ). The schematic is filtered, and only the selected object (Prgm\_Cntr) is displayed.



- Select Options->HDL Analyst Options and click on the Visual Properties tab. Then click an empty Property field, and add the new property to this field and click OK.

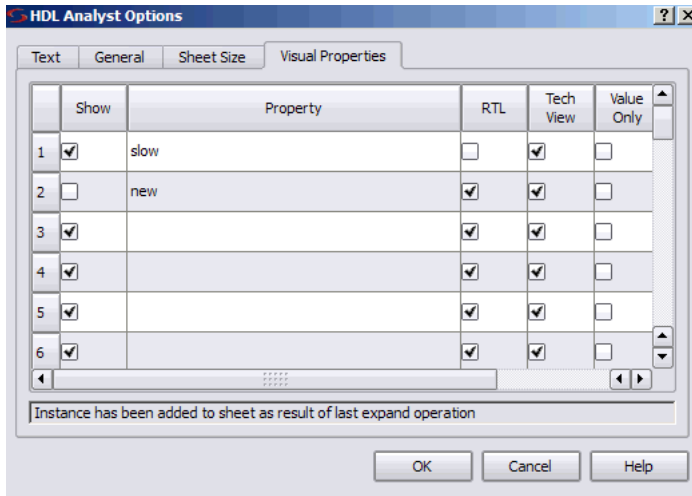


- Make sure that View->Visual Properties is enabled (checked). The new tag appears on any new instance added to the filtered view by subsequent operations.
- To see an expanded view of a pin, click on that pin, right-click to display a menu, and select Expand. The next figure shows an example of an expanded view of a pin.

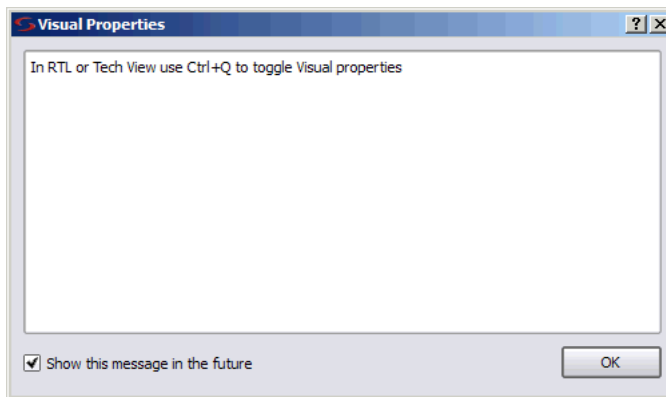



The software expands the connection to the next register and displays it. Because this register is inside `ins_decode`, the software indicates hierarchy with a transparent hierarchical instance (a hollow bounding box surrounding the lower-level logic connection).

- Select Options->HDL Analyst Options->Visual Properties and deselect (uncheck) the Show checkbox next to the new property, and click OK.

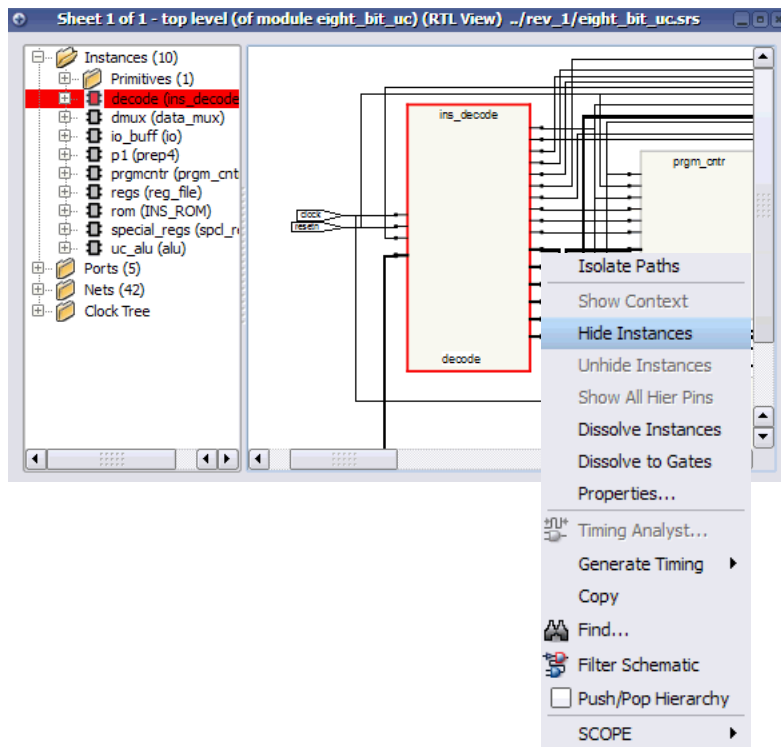


Note, you can also use the shortcut keys Ctrl-q to toggle Visual Properties on or off in the RTL or Technology view as described in the message below.



- Click the Back button () twice to return to the top level.
- 2. Hide an instance in the top-level RTL view.
  - Select the INS\_Decode block, right-click, and select Hide Instances from the pop-up menu.





You see a small H in the lower left corner of the instance, which indicates all lower-level hierarchy is “hidden” from certain operations such as expanding.

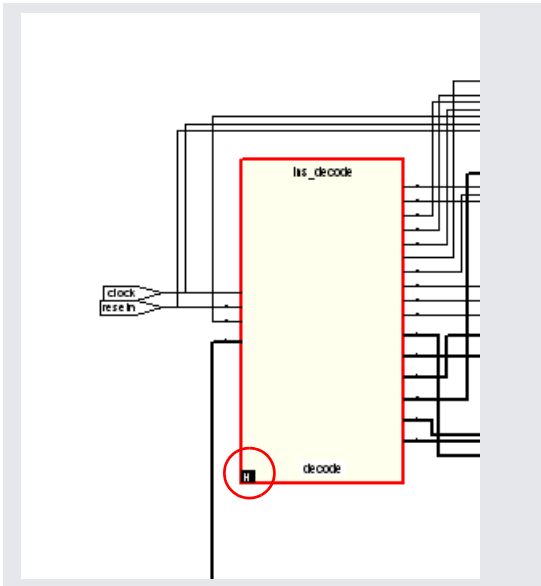

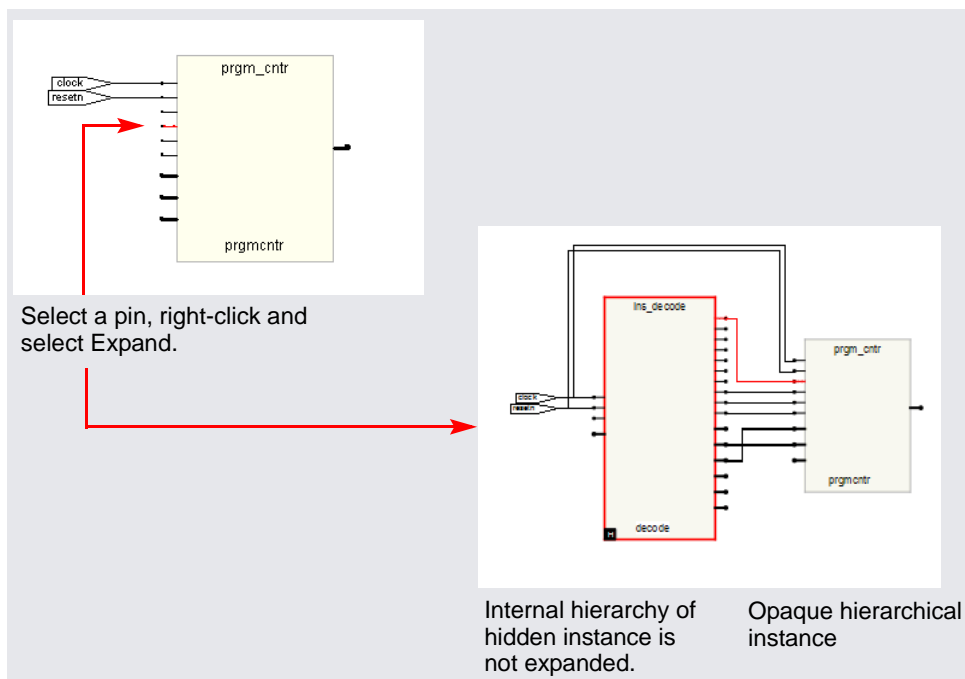



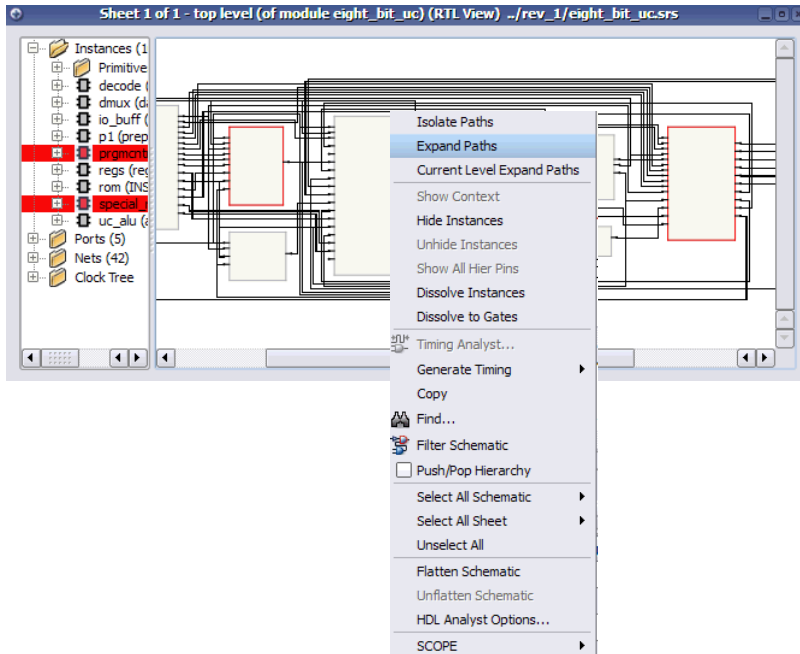
Figure 14: H Indicates a Hidden Instance

- Click in a blank area to deselect everything.
- Select Prgm\_Cntr and press F12 or select the Filter on Selected Gates icon (  ), so that only the selected object (Prgm\_Cntr) is displayed.
- Select the same pin as shown below and in step1, right-click, and select Expand.

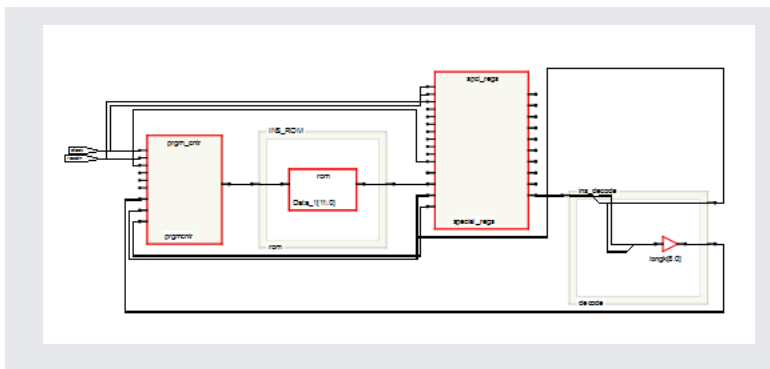
The results are different, because the internal hierarchy of the hidden instance is not expanded.



- Click the Back button (  ) twice to return to the top level.
  - Click the RTL icon and open another window with the top-level RTL view. Zoom in and look at the lower left corner of INS\_Decode block. It is not hidden in this window, although it is hidden in the first RTL window. You can hide different portions of the design hierarchy in different RTL windows.
  - Return to the first RTL window and select INS\_Decode. Right-click and select Unhide Instances. The instance is no longer hidden.
  - Close one of the windows.
3. View the connections between selected instances.
- In the top-level RTL view select Prgm\_Cntr and then, while holding the Ctrl key, select Spcl\_Regs.
  - Right-click and select Expand Paths.



The schematic view displays the hierarchical view between the selected instances, which goes through INS\_ROM.



4. Push into INS\_ROM.

- To push into INS\_ROM, put the Push/Pop Hierarchy cursor over the ROM instance and click. A text file with the ROM data table is displayed.

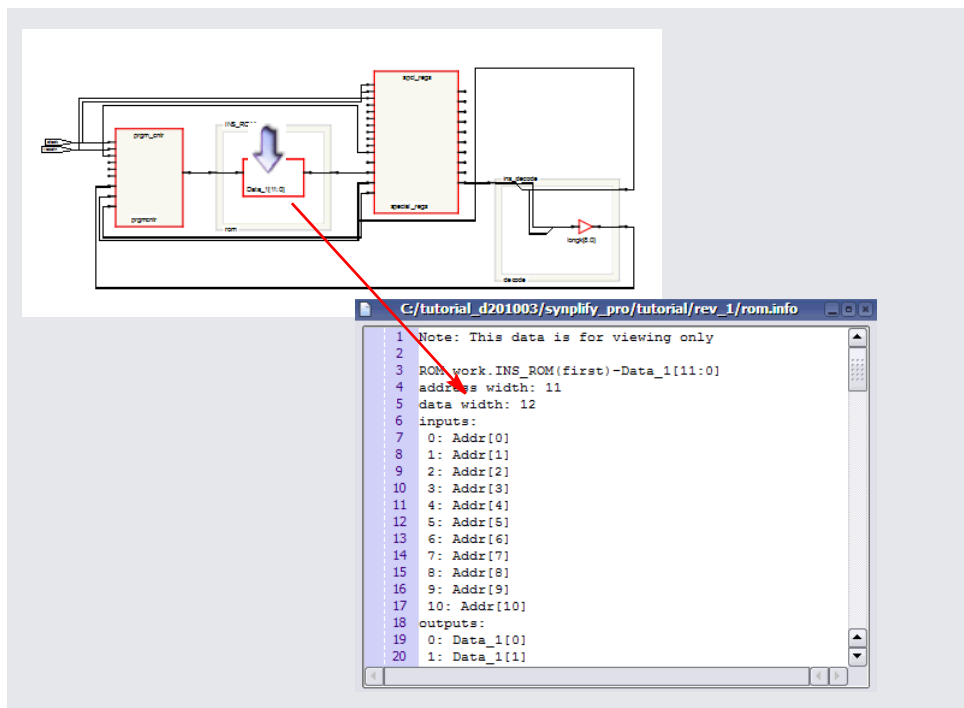


Figure 15: Push into INS ROM

- Close the text window, exit Push/Pop mode, and click the Back button (⏮) until you return to the top level.

#### 5. Flatten hierarchy.

- In the top-level view, right-click and select Flatten Schematic from the pop-up menu. The software flattens the entire design.

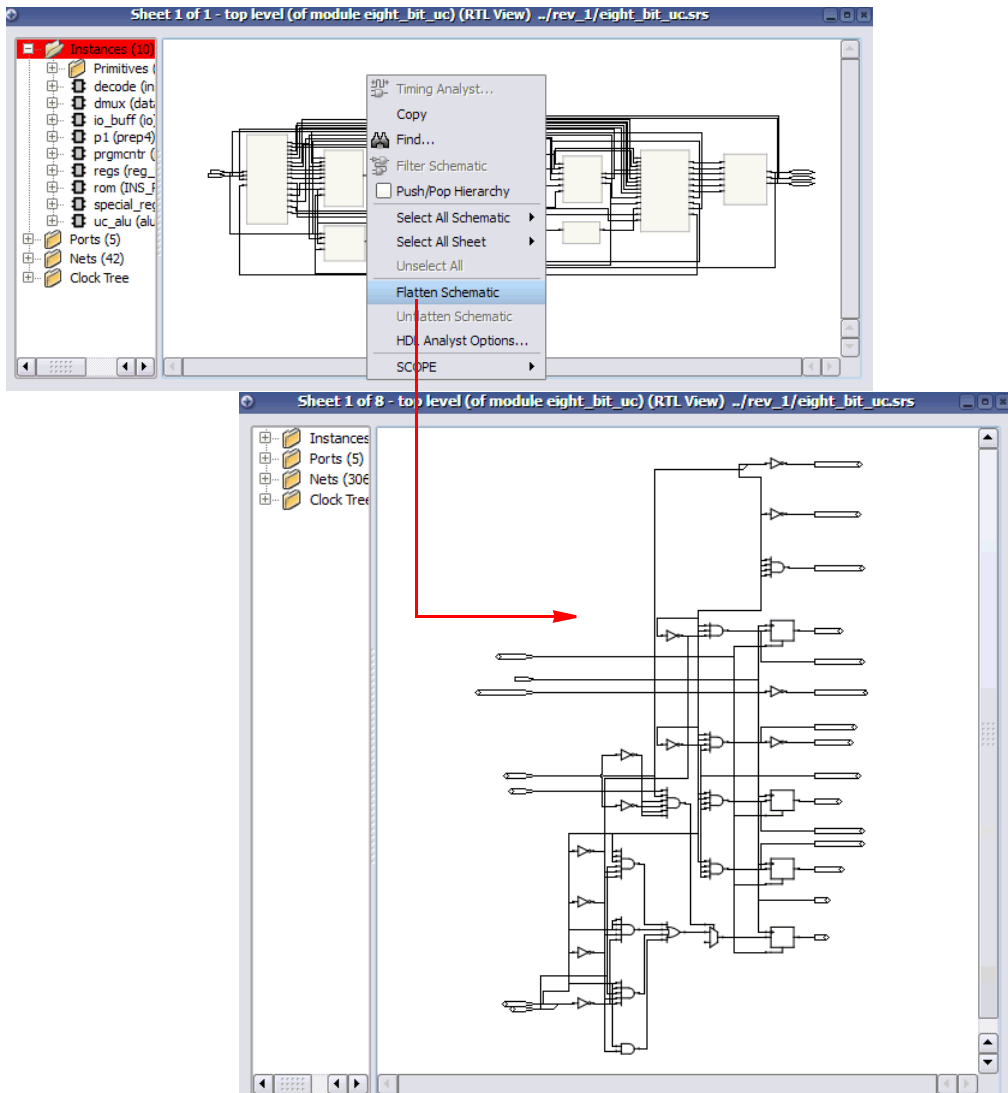
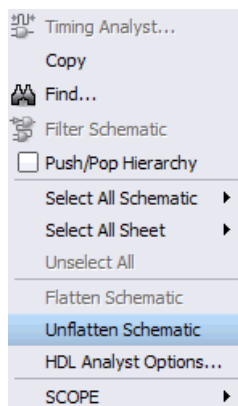


Figure 16: In Top-level RTL View Select Flattened Schematic

- Return to the hierarchical view by right-clicking and selecting UnFlatten Schematic.

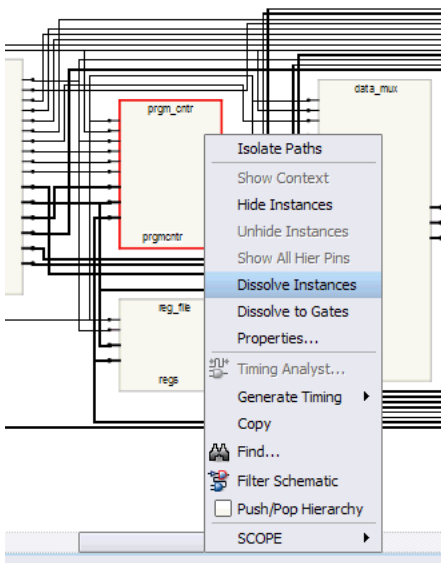


---

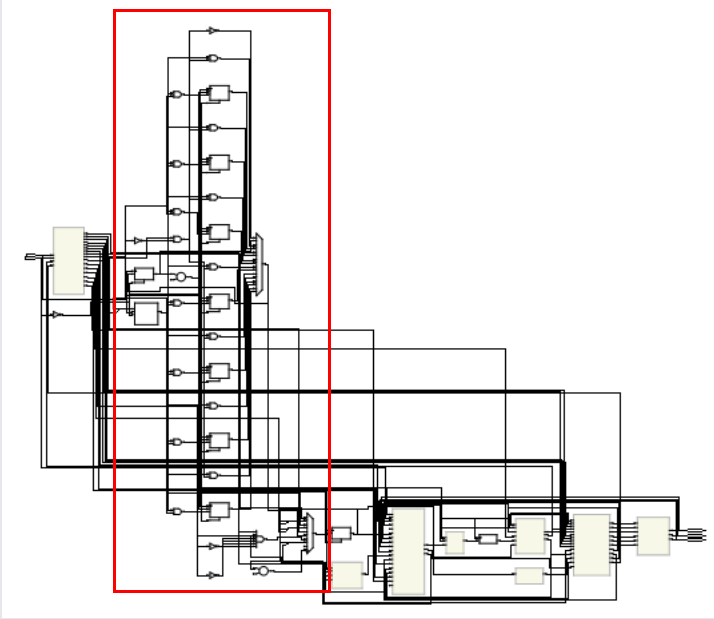
**Note:** You cannot use the Back button, because this is a flattened view, not a filtered view. In a flattened view, there is no history, so Back is not available.


---

- In the unflattened top-level view, select Prgm\_Cntr, right-click and select Dissolve Instances.

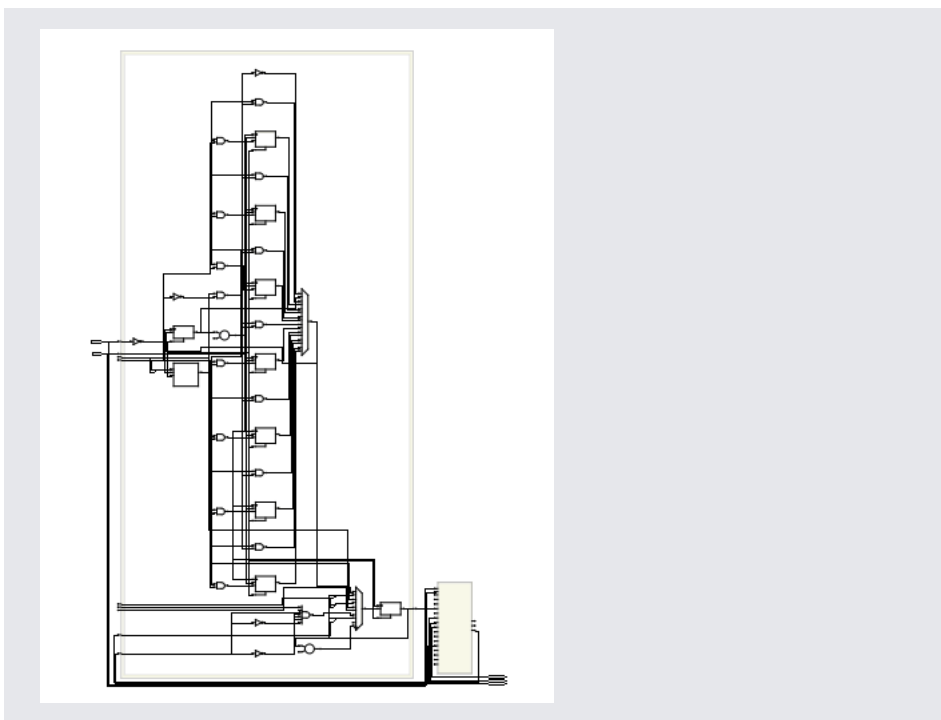



The software flattens the hierarchy for `Prgm_Cntr` only, and displays a flattened view with the internal logic. It retains the hierarchical context of the rest of the design.



- Return to the full, hierarchical view by right-clicking and selecting **Unflatten Schematic** (because this is a flattened view and the **Back** button does not operate). Once you are at the top level, the **Back** button becomes active and you can go back to the previous flattened view.
6. Dissolve hierarchy in a filtered view.
- In the top-level view, select `Prgm_Cntr`, hold down the **Ctrl** key and click on `Data_Mux`. Click the **Filter on Selected Gates** icon (  ) to filter these two instances.
  - In the filtered view, click in a blank area to deselect the instances, then select `Prgm_Cntr`. Right-click and select **Dissolve Instances**. The resulting filtered view shows the internal hierarchy of `Prgm_Cntr` flattened within a transparent instance. `Data_Mux` is not flattened.





- Click the Back button (  ) until you return to the top level.

The Back button works because this is a filtered view, not a flattened view. Filtered views have history.

7. You can minimize the RTL view if you choose or close it.

The rest of the tutorial varies slightly, depending on the technology used. If you do not use the Xilinx vendor, you can follow the methodology used in this flow and substitute device options specific to your vendor.

