Synplicity®

Simply Better Results

Synplify®

Tutorial

October 2002

Synplicity, Inc.
600 West California Avenue
Sunnyvale, CA 94086
(U.S.) +1 408 215-6000 direct
(U.S.) +1 408 222-0263 fax
www.synplicity.com

Preface

Disclaimer of Warranty

Copyright Notice

Trademarks

Restricted Rights Legend

Printed in the U.S.A
October 2002

Synplicity Software License Agreement

Important! READ CAREFULLY BEFORE PROCEEDING

This is a legal agreement between you, the user ("Licensee") and Synplicity, Inc. ("Synplicity") regarding the software program that is attached to or enclosed with this software license agreement, or that is the subject of this documentation (the "SOFTWARE"). The term "SOFTWARE" also includes related documentation (whether in print or electronic form) and, if Licensee is obtaining an update, any pre-existing software and data provided within earlier software releases (to the extent such earlier software and data is retained by, embodied in or in any way used or accessed by the upgraded SOFTWARE provided with this Agreement). If Licensee is a participant in the University Program or has been granted an Evaluation License, then some of the following terms and conditions may not apply (refer to the sections entitled, respectively, **Evaluation License** and **University Program**, below).

By opening the packaging of the SOFTWARE, or by installing or using the SOFTWARE, Licensee agrees to be bound by the terms of this Software License Agreement (the "Agreement"). If Licensee does not agree to the terms of this Agreement, then do not install the SOFTWARE and return the copy of the SOFTWARE to the place from which you obtained it.

Evaluation License. *The following applies, in addition to the other terms and conditions and as a limit on the license, if Licensee has obtained an Evaluation License.* If Licensee has obtained the SOFTWARE pursuant to an evaluation license, then the following additional terms, conditions, and restrictions apply: (a) The license to the SOFTWARE terminates after 20 days (unless otherwise agreed to in writing by Synplicity); and (b) Licensee may use the SOFTWARE only for the sole purpose of tests and other evaluation to determine whether Licensee wishes to license the SOFTWARE on a commercial basis. Licensee shall not use the SOFTWARE to design any integrated circuits for production or pre-production purposes or any other commercial use including, but not limited to, for the benefit of Licensee's customers. If Licensee breaches any of the foregoing restrictions, then Licensee shall pay to Synplicity a license fee equal to Synplicity's standard license fee for the commercial version of the SOFTWARE.

License. Synplicity grants to Licensee a non-exclusive right to install the SOFTWARE and to use or authorize use of the SOFTWARE by up to the number of nodes for which Licensee has a license and for which Licensee has the security key(s) or authorization code(s) provided by Synplicity or its agents. All SOFTWARE must be used within the country for which the systems were licensed and at Licensee's site (contained within a one kilometer radius); however, remote use is permitted by employees who work at the site but are temporarily telecommuting to that same site from less than 50 miles away (for example, an employee who works at a home office on occasion). In addition, Synplicity grants to Licensee a non-exclusive license to copy and distribute internally the documentation portion of the SOFTWARE in support of its license to use the program portion of the SOFTWARE.

Copy Restrictions. This SOFTWARE is protected by United States copyright laws and international treaty provisions and copying not in accordance with this Agreement is forbidden. Licensee may copy the SOFTWARE only as follows: (i) to directly support authorized use under the license and (ii) in order to make a copy of the SOFTWARE for backup purposes. Copies must include all copyright and trademark notices.

Use Restrictions. This SOFTWARE is licensed to Licensee for internal use only. Licensee shall not (and shall not allow any third party to): (i) decompile, disassemble, reverse engineer or attempt to reconstruct, identify or discover any source code, underlying ideas, underlying user interface techniques or algorithms of the SOFTWARE by any means whatever, or disclose any of the foregoing; (ii) provide, lease, lend, or use the SOFTWARE for timesharing or service bureau purposes, on an application service provider basis, or otherwise circumvent the internal use restrictions; (iii) modify, incorporate into or with other software, or create a derivative work of any part of the SOFTWARE; (iv) disclose the results of any benchmarking of the SOFTWARE, or use such results for its own competing software development activities, without the prior written permission of Synplicity; or (v) attempt to circumvent any user limits, maximum gate count limits or other license, timing or use restrictions that are built into the SOFTWARE.

Transfer Restrictions. Licensee shall not sublicense, transfer or assign this Agreement or any of the rights or licenses granted under this Agreement, except in the case of a merger or sale of all or substantially all of Licensee's assets. Permitted transfers, sublicenses or assignments under this Section continue to be subject to the rules regarding Export of the SOFTWARE.

Ownership of the SOFTWARE. Synplicity retains all right, title, and interest in the SOFTWARE (including all copies), and reserves all rights not expressly granted to Licensee. This License is not a sale of the original SOFTWARE or of any copy.

Ownership of Design Techniques. "Design" means the representation of an electronic circuit or device(s), derived or created by Licensee through the use of the SOFTWARE in its various formats, including, but not limited to, equations, truth tables, schematic diagrams, textual descriptions, hardware description languages, and netlists. "Design Techniques" means the Synplicity-supplied data, circuit and logic elements, libraries, algorithms, search strategies, rule bases, and technical information incorporated in the SOFTWARE and employed in the process of creating Designs. Synplicity retains all right, title and interest in and to Design Techniques incorporated into the SOFTWARE, including all intellectual property rights embodied therein. Licensee acknowledges that Synplicity is in the business of licensing SOFTWARE which incorporates Design Techniques. Licensee agrees that in the event Licensee voluntarily discloses any design techniques to Synplicity without designating such as Licensee's Confidential Information, Synplicity has an unrestricted, royalty-free right to incorporate those Design Techniques into its software, documentation and other products, and to sublicense third parties to use those incorporated design techniques.

Protection of Confidential Information. "Confidential Information" means (i) the source code of the SOFTWARE, and any included trade secrets (including any technology, idea, algorithm or information contained in the SOFTWARE, and specifically including Design Techniques); (ii) either party's product plans, designs, costs, prices and names; non-published financial information; marketing plans; business opportunities; personnel; research; development or know-how; (iii) any information designated by the disclosing party as confidential in writing or, if disclosed orally, designated as confidential at the time of disclosure and reduced to writing and given to the receiving party and designated as confidential in writing within 30 days; and (iv) the terms and conditions of this Agreement; provided, however that "Confidential Information" will not include information that: (a) is or becomes generally known or available by publication, commercial use or otherwise through no fault of the receiving party; (b) is known and has been reduced to tangible form by the receiving party at the time of disclosure and is not subject to restriction; (c) is independently developed by the receiving party without use of the disclosing party's Confidential Information; (d) is lawfully obtained from a third party who has the right to make such disclosure; or (e) is released for publication by the disclosing party in writing.

Each party will protect the other's Confidential Information from unauthorized dissemination and use with the same degree of care that each such party uses to protect its own like information. Neither party will use the other's Confidential Information for purposes other than those necessary to directly further the purposes of this Agreement. Neither party will disclose to third parties the other's Confidential Information without the prior written consent of the other party.

Termination. Synplicity may terminate this Agreement in the event of breach or default by Licensee. Upon termination Licensee will relinquish all rights under this Agreement, and must cease using the SOFTWARE and return or destroy all copies (and partial copies) of the SOFTWARE and documentation.

Export. Licensee warrants that it is not prohibited from receiving the SOFTWARE under U.S. export laws; that it is not a national of a country subject to trade sanctions; that it will not use the SOFTWARE in a location that is the subject of U.S. trade sanctions that would cover the SOFTWARE and that to its knowledge it is not on the U.S. Department of Commerce, Bureau of Export Administration's Entity List or otherwise prohibited from obtaining goods of this sort from the United States. Licensee shall not allow the Synplicity SOFTWARE to be sent or used in any country except in compliance with applicable U. S. laws and regulations.

Limited Warranty and Disclaimer. Synplicity warrants that the program portion of the SOFTWARE will perform substantially in accordance with the accompanying documentation for a period of 90 days from the date of receipt. Synplicity's entire liability and Licensee's exclusive remedy for a breach of the preceding limited warranties shall be, at Synplicity's option, either (a) return of the license fee, or (b) providing a fix, patch, work-around, or replacement of the SOFTWARE that does not meet such limited warranty. In either case, Licensee must return the SOFTWARE to Synplicity with a copy of the purchase receipt or similar document. Replacements are warranted for the remainder of the original warranty period or 30 days, whichever is longer. Some states/jurisdictions do not allow limitations on duration of an implied warranty, so the above limitation may not apply. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES OR CONDITIONS, EITHER EXPRESS OR IMPLIED, ARE MADE BY SYNPLICITY WITH RESPECT TO THE SOFTWARE AND THE ACCOMPANYING DOCUMENTATION (STATUTORY OR OTHERWISE), AND SYNPLICITY EXPRESSLY DISCLAIMS ALL WARRANTIES AND CONDITIONS NOT EXPRESSLY STATED HEREIN, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, NONINFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE. SYNPLICITY DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET LICENSEE'S REQUIREMENTS, BE UNINTERRUPTED OR ERROR FREE, OR THAT ALL DEFECTS IN THE PROGRAM WILL BE CORRECTED. Licensee assumes the entire risk as to the results and performance of the SOFTWARE. Some states/jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply.

Limitation of Liability. IN NO EVENT SHALL SYNPLICITY OR ITS AGENTS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL OR INCIDENTAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTIONS, LOSS OF BUSINESS INFORMATION, OR OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THESE SYNPLICITY PRODUCTS, EVEN IF SYNPLICITY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Synplicity be liable to Licensee for damages in an amount greater than the fees paid for the use of the SOFTWARE. Some states/jurisdictions do not allow the limitation or exclusion of incidental or consequential damages, so the above limitations or exclusions may not apply.

Intellectual Property Right Infringement. If a claim alleging infringement of an intellectual property right arises concerning the SOFTWARE (including but not limited to patent, trade secret, copyright or trademark rights), Synplicity in its sole discretion may elect to defend or settle such claim. Synplicity in the event of such a claim may also in its sole discretion elect to terminate this Agreement and all rights to use the SOFTWARE, and require the return or destruction of the SOFTWARE, with a refund of the fees paid for use of the SOFTWARE less a reasonable allowance for use and shipping.

Export. Licensee warrants that it is not prohibited from receiving the SOFTWARE under U.S. export laws; that it is not a national of a country subject to trade sanctions; that it will not use the SOFTWARE in a location that is the subject of U.S. trade sanctions that would cover the SOFTWARE and that to its knowledge it is not on the U.S. Department of Commerce, Bureau of Export Administration's Entity List or otherwise prohibited from obtaining goods of this sort from the United States.

Miscellaneous. If Licensee is a corporation, partnership or similar entity, then the license to the SOFTWARE that is granted under this Agreement is expressly conditioned upon acceptance by a person who is authorized to sign for and bind the entity. This Agreement is the entire agreement between Licensee and Synplicity with respect to the license to the SOFTWARE, and supersedes any previous oral or written communications or documents (including, if you are obtaining an update, any agreement that may have been included with the initial version of the SOFTWARE). This Agreement is governed by the laws of the State of California, USA. This Agreement will not be governed by the U. N. Convention on Contracts for the International Sale of Goods and will not be governed by any statute based on or derived from the Uniform Computer Information Transactions Act (UCITA). If any provision of this Agreement is found to be invalid or unenforceable, it will be enforced to the extent permissible and the remainder of this Agreement will remain in full force and effect. Failure to prosecute a party's rights with respect to a default hereunder will not constitute a waiver of the right to enforce rights with respect to the same or any other breach.

Government Users. The Software contains commercial computer software and commercial computer software documentation. In accordance with FAR 12.212 and DFARS 227.7202, use, duplication or disclosure is subject to restrictions under paragraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at 252.227-7013, and further restricted by this Agreement. Synplicity, Inc., 600 W. California Ave, Sunnyvale, CA 94086, U. S. A.

SCF Reader. *The following terms and conditions apply only to the SCF reader, which provides the functionality to support input of designs in SCF format (the "Reader"), and which may be included with the Software.* Such terms are in addition to the other terms in this Agreement. **With respect to the Reader, Synplicity disclaims, on its own behalf and on behalf of any of its licensors or suppliers who have contributed to the Reader, any and all warranties and conditions, express or implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability and fitness for a particular purpose.** Synplicity hereby excludes, on its own behalf and on behalf of any of its licensors or suppliers who have contributed to the Reader, any and all liability for damages, including direct, indirect, special, incidental and consequential damages, such as lost profits. The foregoing applies notwithstanding the type of claim (whether contract, tort or otherwise) and notwithstanding the failure of essential purpose of any limited remedy. If there are any terms in this Agreement regarding the Reader that differ from those terms on which Synplicity obtained the Reader, then those terms are offered by Synplicity and not by any other party. If you would like a source code version of the Reader, please contact Synplicity at support@synplicity.com, and Synplicity will make it available on reasonable terms.

University Program. *The following section applies only if Licensee is a participant in Synplicity's University Program; it does not replace the remainder of the Agreement and supersedes only those terms that directly conflict.*

University Program: License. Subject to the terms and conditions of this Agreement, Synplicity hereby grants to Licensee (a University) for the License Term (defined below), a non-exclusive license, only for purposes of course work or teaching in connection with a university-sponsored class, or for academic research either sponsored by or conducted under the auspices of Licensee, to (a) install and use the SOFTWARE, and (b) reproduce and distribute copies of the documentation included in the SOFTWARE subject only to payment for those copies (which may be based on the number of users, the number and type of copies, or both). If the SOFTWARE is licensed pursuant to a node-locked license, then the Licensee may install and use the SOFTWARE on the authorized workstations. If the SOFTWARE is licensed pursuant to a floating license, then the Licensee may install the SOFTWARE on the authorized server and use the SOFTWARE on up to the number of nodes for which Licensee has paid license fees and Synplicity has granted authorization.

University Program: License Term and Termination. For purposes of the University Program, "License Term" means one year unless otherwise agreed to in writing. This Agreement will terminate at the end of the License Term, unless earlier terminated in accordance with this Agreement.

University Program: License Restrictions. As Licensee, University may not (i) allow access to the SOFTWARE by any user not registered for a course or participating in an academic research project for which use of the SOFTWARE has been authorized; (ii) use the SOFTWARE to design

Contents

Synplify Tutorial

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

Introduction to the Tutorial

This tutorial is designed to walk you through some typical tasks and familiarize you with the user interface, but it does not cover all the possible tasks you could do. For additional information, refer to the following sources:

For information about...	See...
Installation information	The installation instructions
Information about the interface	*Synplify Reference Manual*
Common tasks not covered in the tutorial	*Synplify User Guide*
Advanced techniques	*Synplify User Guide*
Detailed information about the commands and syntax	*Synplify Reference Manual*

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

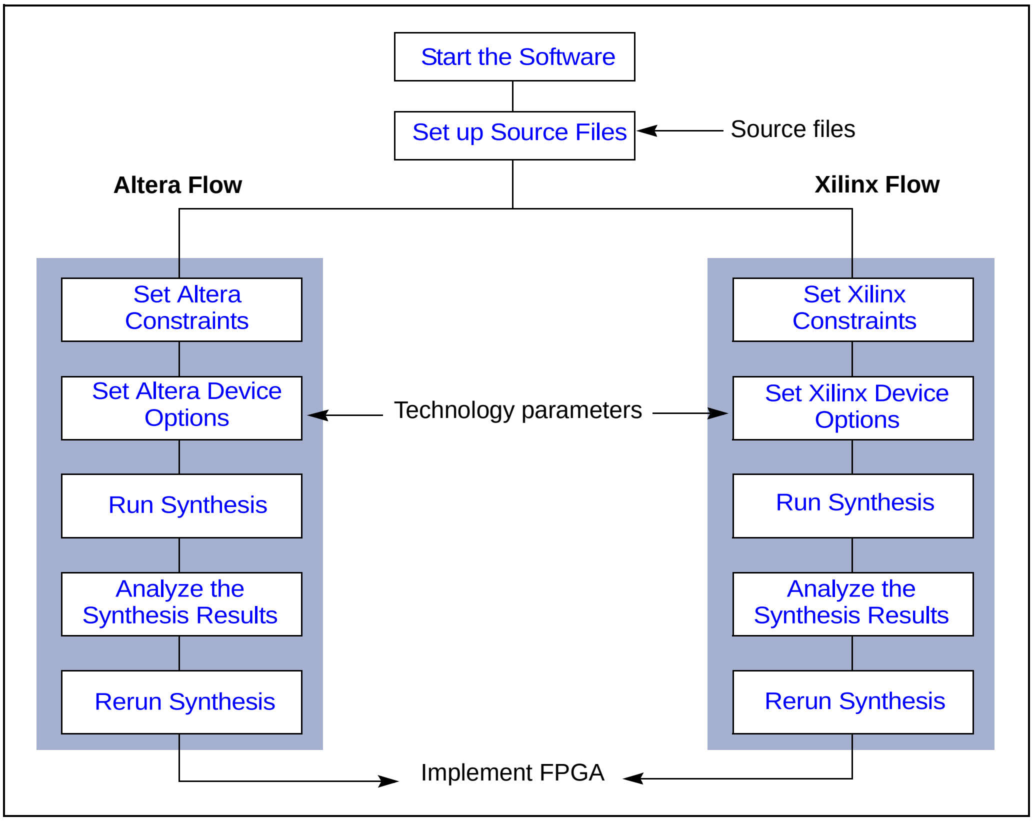
This tutorial assumes that you have

- Installed the software correctly
- Obtained the necessary licenses

If you need help with these issues, refer to the installation instructions.

The Tutorial Design Flow

This flow diagram graphically illustrates the procedures demonstrated in this tutorial:

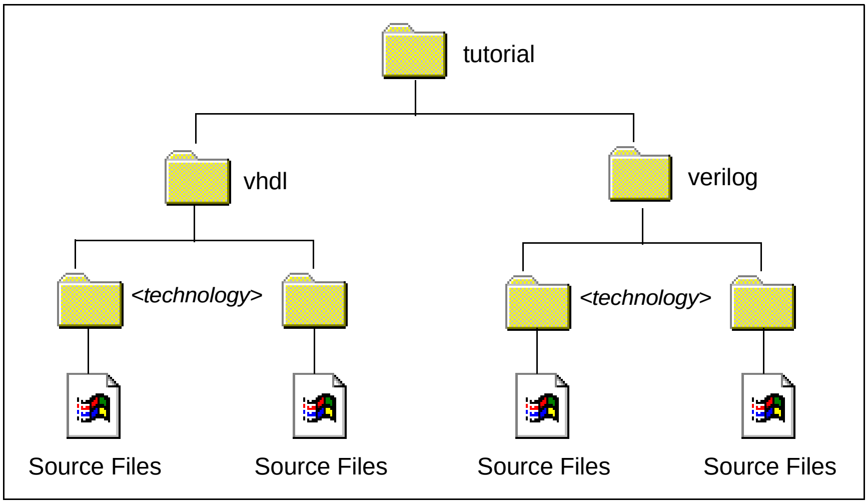


Start the Software

You can start the software and run the tutorial from a PC, Linux, or UNIX workstation. If your setup is different, talk to your system administrator.

1. Make sure you have installed the software and obtained a license.
2. Copy the tutorial directory to your working area. Keep the directory structure, because the tutorial is based on this structure. When you work on your own designs, you can set up the structure as you want.

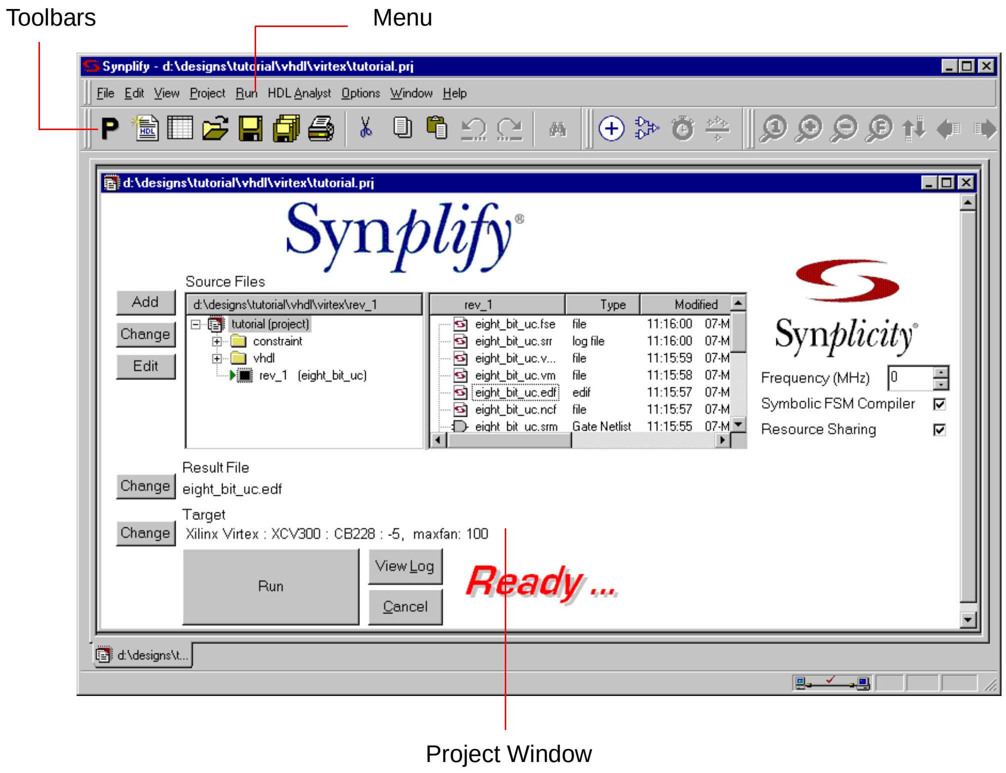
   Your tutorial directory contains the following directories and files. The *<technology>* directories are altera and xilinx, the two technologies used in this tutorial.



3. Make sure you have read and write privileges for the files.
4. Start the software.
   - On a PC, choose the current release of the software: Start->Programs->Synplicity->Synplify 7.2.
   - On a UNIX workstation, type this at the command line:

     ```
     synplify
     ```

     Either command starts the synthesis tool. If you have run the software before, the window displays the previous project. You see the window shown in the following figure. For more information about the interface, see the *Synplify Reference Manual*.


Toolbars
Menu
Synplify - d:\designs\tutorial\vhdl\virtex\tutorial.prj
File Edit View Project Run HDL Analyst Options Window Help
d:\designs\tutorial\vhdl\virtex\tutorial.prj
Synplify
Synplicity
Add
Change
Edit
Run
View Log
Cancel
Ready ...
d:\designs\t...
Project Window

Source Files

d:\designs\tutorial\vhdl\virtex\rev_1
- tutorial (project)
  - constraint
  - vhdl
  - rev_1 (eight_bit_uc)

rev_1	Type	Modified	
eight_bit_uc.fse	file	11:16:00	07-M
eight_bit_uc.srr	log file	11:16:00	07-M
eight_bit_uc.v...	file	11:15:59	07-M
eight_bit_uc.vm	file	11:15:58	07-M
eight_bit_uc.edf	edif	11:15:57	07-M
eight_bit_uc.ncf	file	11:15:57	07-M
eight_bit_uc.srm	Gate Netlist	11:15:55	07-M

Frequency (MHz) 0
Symbolic FSM Compiler ✓
Resource Sharing ✓

Result File
eight_bit_uc.edf

Target
Xilinx Virtex : XCV300 : CB228 : -5, maxfan: 100

You can access common commands in different ways: through the menu, popup menus, keyboard shortcuts, and icons. The tutorial uses different methods to access the commands.

Set up Source Files

The first step is to set up the input files for the project. A project is a design that you work on, and consists of the input files and the implementation files. This section shows you how to set up a project file, handle warnings, and do some typical analysis operations with the HDL Analyst tool. The information is organized as follows:

- Create a Project File, next
- Check the Source Files on page 9
- Resolve Source File Warnings on page 10
- Examine the RTL View and Navigate Hierarchy on page 11
- Find and Crossprobe on page 15
- Filter, Expand, Hide, and Dissolve on page 18

Create a Project File

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

1. In the window, select File -> Build Project, which opens a dialog box.
2. Do the following in the Select Files to Add to Project dialog box to set the directory for the source files and the type of files:
   - Set the Look In: field at the top of the form to the appropriate directory: `tutorial/vhdl/<`*`technology`*`>`. If you plan to work through the Altera flow in the tutorial, select altera for the technology; if you want to work through the Xilinx flow, select xilinx. This tutorial only works through these two technologies, but the software supports many other technologies. If you use another technology, you can still follow the general flow of the tutorial and substitute parameters specific to your technology.
   - Check that Files of Type is set to HDL Files (*.vhd, *.v). This tutorial uses only VHDL source files. If you want to use Verilog, use the Verilog source files provided in the `tutorial/verilog` directory, but some of

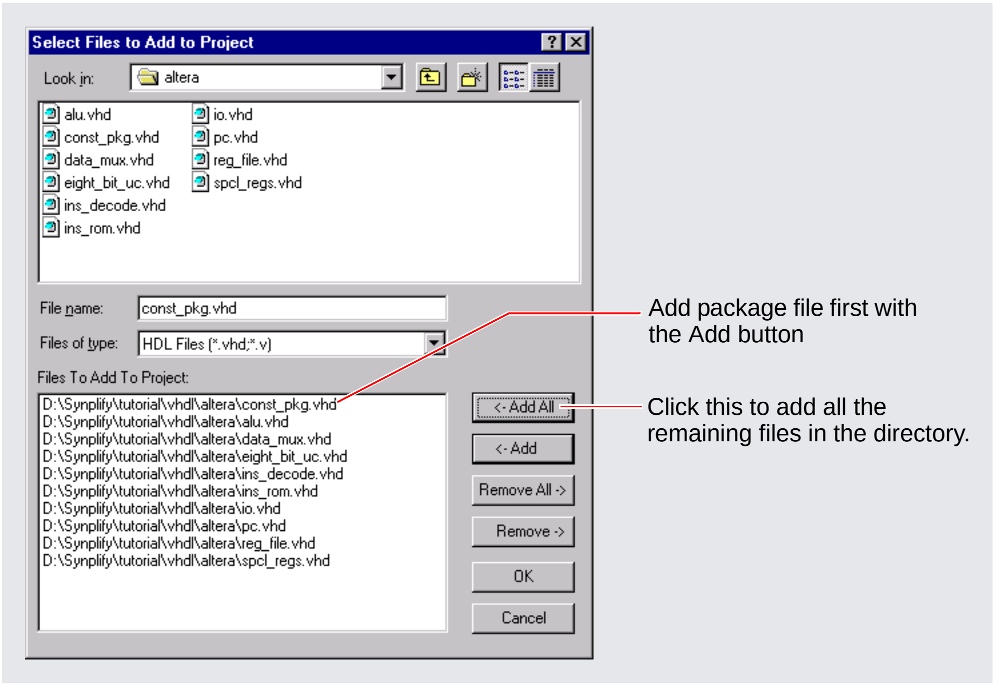
the steps might not exactly match the steps documented in this tutorial. For your own designs, you can run the software with VHDL or Verilog source files.

The next step is to add the source files to your project file.

3. Add the source files, by doing the following in the dialog box:

   – Add the VHDL package file first: select the `const_pkg.vhd` file and click Add on the right side of the form. For Verilog, you do not need to do this. The file appears in the text box at the bottom of the form.

   – Add the other files. Click the Add All button on the right side of the form to add all the files in the directory. The text box in the lower half of the dialog box now lists all the .vhd files in the directory.



```
D:\Synplify\tutorial\vhdl\altera\const_pkg.vhd
D:\Synplify\tutorial\vhdl\altera\alu.vhd
D:\Synplify\tutorial\vhdl\altera\data_mux.vhd
D:\Synplify\tutorial\vhdl\altera\eight_bit_uc.vhd
D:\Synplify\tutorial\vhdl\altera\ins_decode.vhd
D:\Synplify\tutorial\vhdl\altera\ins_rom.vhd
D:\Synplify\tutorial\vhdl\altera\io.vhd
D:\Synplify\tutorial\vhdl\altera\pc.vhd
D:\Synplify\tutorial\vhdl\altera\reg_file.vhd
D:\Synplify\tutorial\vhdl\altera\spcl_regs.vhd
```

   – Click OK.

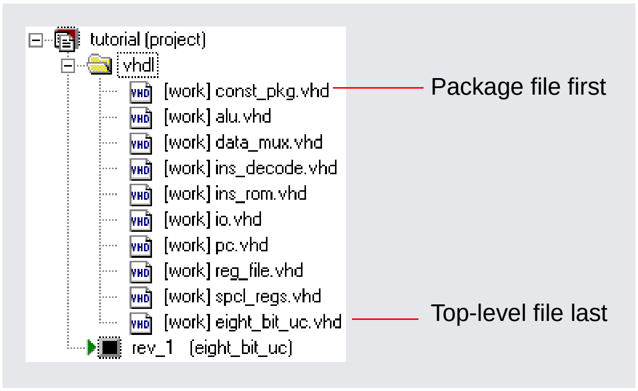
   Your project window displays a newly-created project file called const_pkg with a folder called vhdl and a directory under it called rev_1, which represents the first implementation of your design. Implementations are revisions of your design within the context of the synthesis software, and do not replace external source code control software and processes. Each implementation has its own synthesis and device options and its own project-related files.

– If your files are not in a `vhdl` folder under the `const_pkg` directory, set this preference by selecting the Options->Project View Options command and checking the View project files in folders box. This separates one kind of file from another in the Project view by putting them in separate folders.

4. Click on the plus sign next to the vhdl folder. You see a list of the VHDL source files that you added for the project.

5. Make sure the package file (`const_pkg.vhd`) is the first file and the top-level file (`eight_bit_uc.vhd`) is the last in the list of files in the Project window. To move the top-level file, click `eight_bit_uc.vhd` and drag it to the end of the list of files. The order of the rest of the files does not matter.

6. Save and rename the project. Select File->Save, type tutorial as the name of the project, and click Save. The Project window reflects your changes.



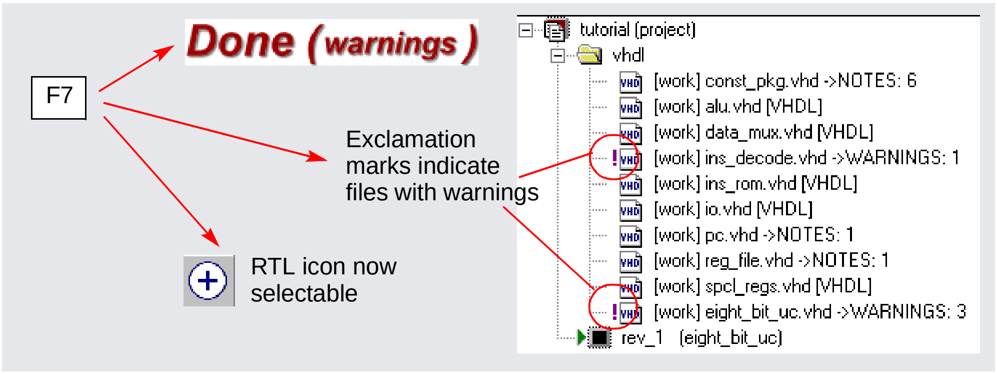
```
tutorial (project)
  vhdl
    [work] const_pkg.vhd
    [work] alu.vhd
    [work] data_mux.vhd
    [work] ins_decode.vhd
    [work] ins_rom.vhd
    [work] io.vhd
    [work] pc.vhd
    [work] reg_file.vhd
    [work] spcl_regs.vhd
    [work] eight_bit_uc.vhd
  rev_1  (eight_bit_uc)
```

Check the Source Files

For the purposes of this tutorial, you compile and check the source files as a separate step. Normally, compilation is part of synthesis, when you press Run. If you are using Verilog files, the specifics described here will differ but you get the same error.

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

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

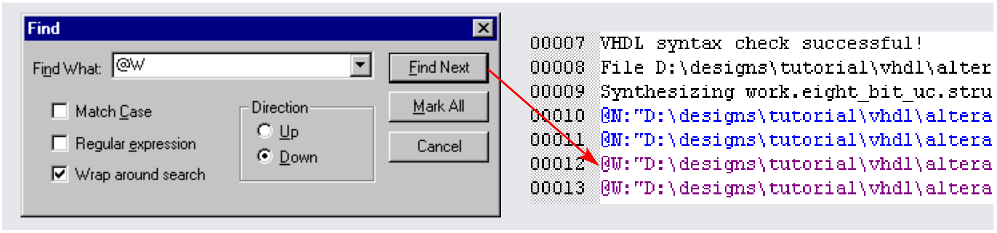


```
tutorial (project)
  vhdl
    [work] const_pkg.vhd ->NOTES: 6
    [work] alu.vhd [VHDL]
    [work] data_mux.vhd [VHDL]
  ! [work] ins_decode.vhd ->WARNINGS: 1
    [work] ins_rom.vhd [VHDL]
    [work] io.vhd [VHDL]
    [work] pc.vhd ->NOTES: 1
    [work] reg_file.vhd ->NOTES: 1
    [work] spcl_regs.vhd [VHDL]
  ! [work] eight_bit_uc.vhd ->WARNINGS: 3
  rev_1   (eight_bit_uc)
```

2. Review the warnings next to the `ins_decode.vhd` and `eight_bit_uc.vhd` files by doing the following:

   – Click View Log to open the log file.

   – Press Ctrl-f to open the Find dialog box. Enter @W as the search criteria, and click Find Next. The pointer moves to the first warning in the log file. All warnings are in purple.



Find What: @W
[ ] Match Case
[ ] Regular expression
[✓] Wrap around search
Direction: Up [✓ Down]

```
00007 VHDL syntax check successful!
00008 File D:\designs\tutorial\vhdl\alter
00009 Synthesizing work.eight_bit_uc.stru
00010 @N:"D:\designs\tutorial\vhdl\altera
00011 @N:"D:\designs\tutorial\vhdl\altera
00012 @W:"D:\designs\tutorial\vhdl\altera
00013 @W:"D:\designs\tutorial\vhdl\altera
```

   Click Find Next again to go to the next warning. Review all warnings and then click Cancel in the Find dialog box. Close the log file window.

You can now deal with the warning messages, as described in the following section.

Resolve Source File Warnings

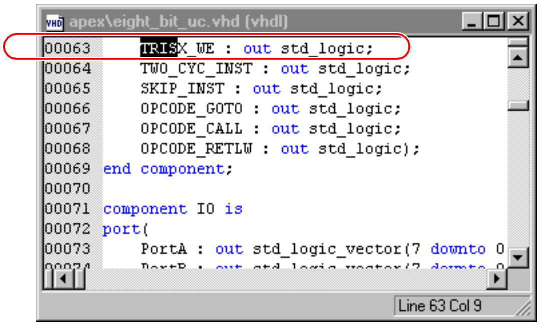
You can edit the source files directly in the Text Editor window.

1. Double-click the ins_decode.vhd source file and check the file:
   - The Text Editor window opens the source file at the line with the error. If you are not at the line with the warning, press F5 to go to the correct line. Read the highlighted line. The `TRIS_WE : out std_logic` line is highlighted.
   - Check the bottom of the Text Editor window for an explanation of the error:
     `Warning: Map for port tris_we of component ins_decode not found.`
   - Close the window with the ins_decode.vhd source file.
2. Return to the project view and check the top-level design file:
   - Double-click eight_bit_uc.vhd in the project window to open the top-level file. You should be at the following line with the error:

     ```
     TRISX_WE : out std_logic;
     ```

     There is a discrepancy between port definitions at the component level and the top level.

     If you are not at the line with the error, go to the beginning of the file and type Ctrl-f. Use TRIS as the search criteria and click Find Next until you find the line shown below.



```
        TRISX_WE : out std_logic;
        TWO_CYC_INST : out std_logic;
        SKIP_INST : out std_logic;
        OPCODE_GOTO : out std_logic;
        OPCODE_CALL : out std_logic;
        OPCODE_RETLW : out std_logic);
end component;

component IO is
port(
        PortA : out std_logic_vector(7 downto 0
```

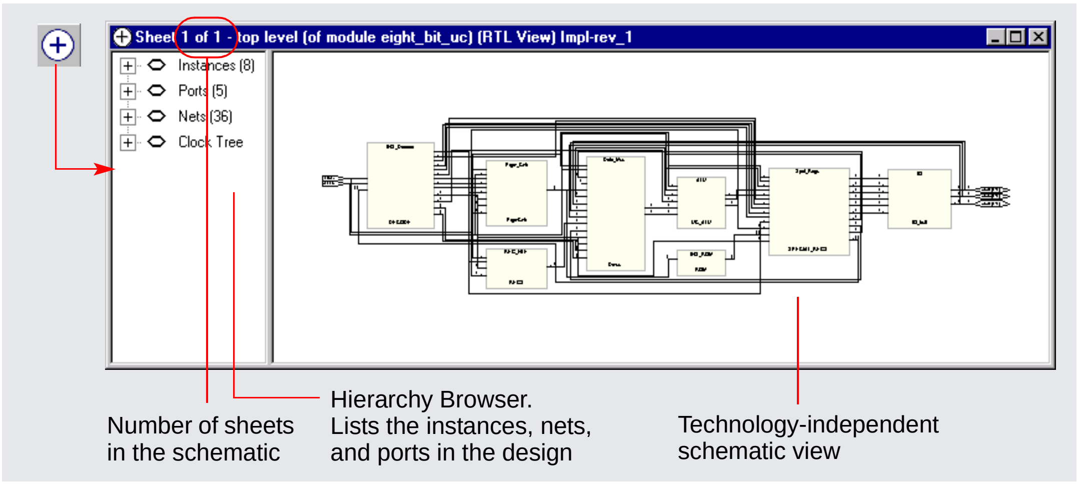
3. Correct the error by changing `TRISX_WE` to `TRIS_WE` to match the port defined in the `ins_decode` component. Normally you would also check the schematic view of the design, but for the purposes of the tutorial, just correct the typing mistake.
4. Close the editing window, and click Yes in the dialog box that asks you if you want to save your changes.
5. Press F7 to recompile.

   This time there are no errors, and the status box displays this message: Done!

Examine the RTL View and Navigate Hierarchy

Now that you have compiled the design and your source files do not have any errors, you can look at an RTL schematic view of the design. You must have the HDL Analyst option to run this part of the tutorial. This section covers basic zooming and hierarchy navigation; *Find and Crossprobe* on page 15 and *Filter, Expand, Hide, and Dissolve* on page 18 discuss some other analysis techniques. Synplicity'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.

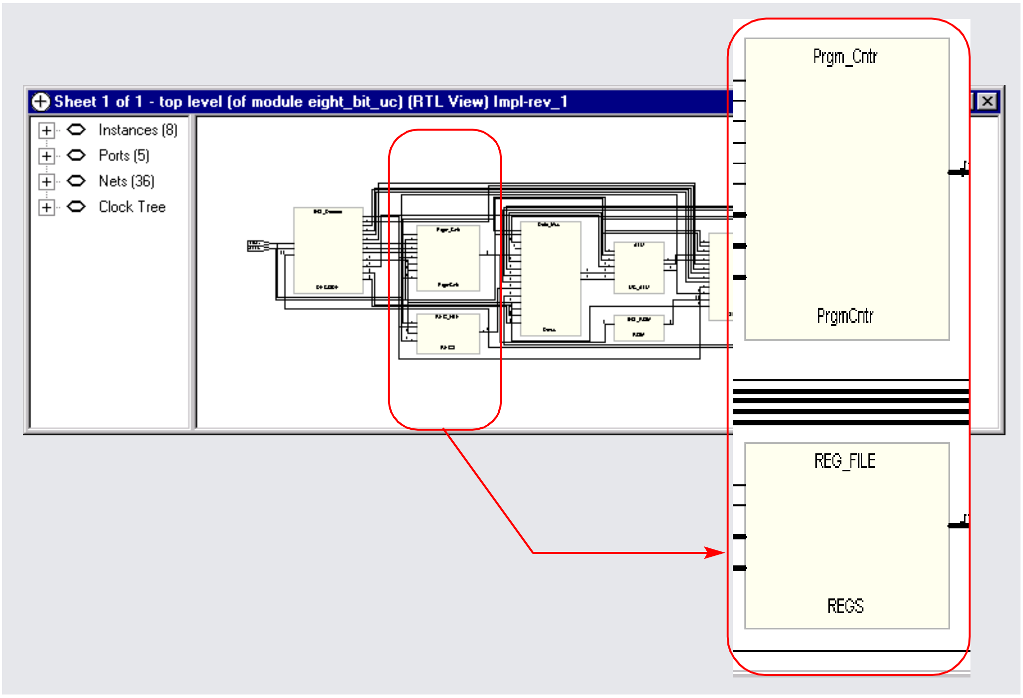
1. Click the RTL View icon from the toolbar (⊕), or select HDL Analyst -> RTL -> Hierarchical View to open the RTL view. If pin names are displayed, reduce clutter in the design by selecting Options->Schematic Options and disabling the Show Pin Names option. Your schematic should look like the one shown in the following figure.


Sheet 1 of 1 - top level (of module eight_bit_uc) (RTL View) Impl-rev_1
Instances (8)
Ports (5)
Nets (36)
Clock Tree
Number of sheets in the schematic
Hierarchy Browser. Lists the instances, nets, and ports in the design
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 pieces of logic from the source code.

2. To view the design, use these icons from the toolbar, the mouse strokes (see Help->Mouse Stroke Tutor), or the corresponding commands from the View menu.

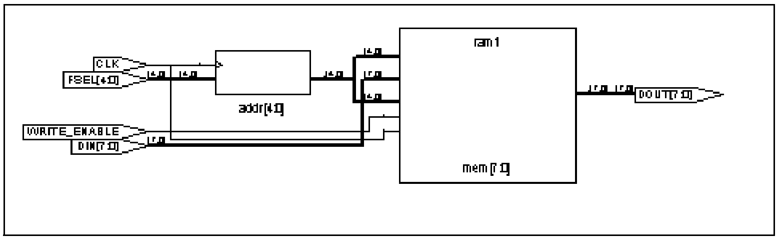
   – Zoom into the area shown in the following figure by clicking the lens icon with a plus (+) and clicking the Z-shaped zoom cursor over the area. Click as many times as needed to get a magnification level that is comfortable. You can also zoom by clicking and dragging the Z-shaped cursor diagonally to specify a rectangular area for zooming, or by pressing Alt and drawing a diagonal mouse stroke from upper right to lower left over the area to be zoomed. On some platforms, Alt functionality is mapped to another key, and you must use the appropriate key to use mouse strokes.


Sheet 1 of 1 - top level (of module eight_bit_uc) (RTL View) Impl-rev_1
Instances (8)
Ports (5)
Nets (36)
Clock Tree
Prgm_Cntr
PrgmCntr
REG_FILE
REGS

– Exit Zoom mode by clicking on the Zoom in icon again, or by right-clicking. The Z-shaped zoom cursor changes back to the default crosshair selection cursor.

3. Select the Push/Pop Hierarchy icon ( ). The cursor changes to a double-headed arrow with an X through it when it is over areas of the design without underlying hierarchy. When it is over a component that has hierarchy below it, it changes to an arrow pointing downward.

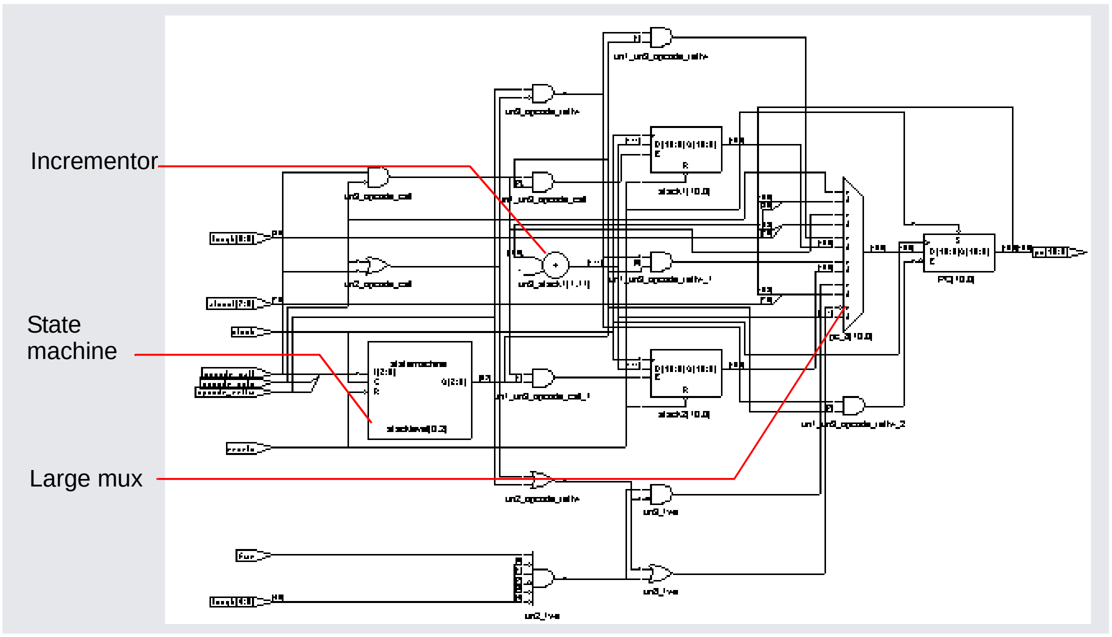
   – Click on the REGS block to push down into it. See the lower-level hierarchy and how the software infers the RAM.



– Pop up to the top level by clicking in an empty area (the cursor changes to an upward-pointing arrow). Alternatively, use the mouse stroke: press Alt (or the key to which Alt functionality is mapped), 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 . Return to the top-level view by clicking the Fwd icon ().

– Click on the Prgm_Cntr block and push down into it. To use a mouse stroke for pushing, put your cursor over the block, press Alt, and draw a vertical line going downwards. In the schematic view, note the abstracts used to represent high-level behavior: incrementor, state machine, and large mux.



– Return to the top level and 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 *Examine the RTL View and Navigate Hierarchy* on page 11 and *Filter, Expand, Hide, and Dissolve* on page 18.

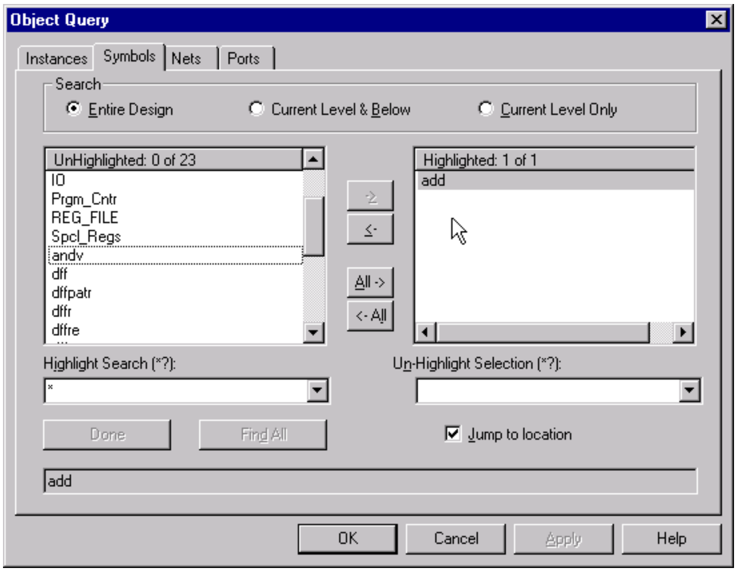
1. With the top-level RTL view open, type Ctrl-f or select Edit->Find.

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

2. Do the following:

   – Click the Symbols tab and set the search range to Entire Design.

   – Scroll down in the Unhighlighted box on the left and double-click add. This moves add to the Highlighted box on the right. Click OK.



Object Query

Instances | Symbols | Nets | Ports

Search: [✓ Entire Design] Current Level & Below Current Level Only

UnHighlighted: 0 of 23
IO
Prgm_Cntr
REG_FILE
Spcl_Regs
andv
dff
dffpatr
dffr
dffre

Highlighted: 1 of 1
add

Highlight Search (*?): *

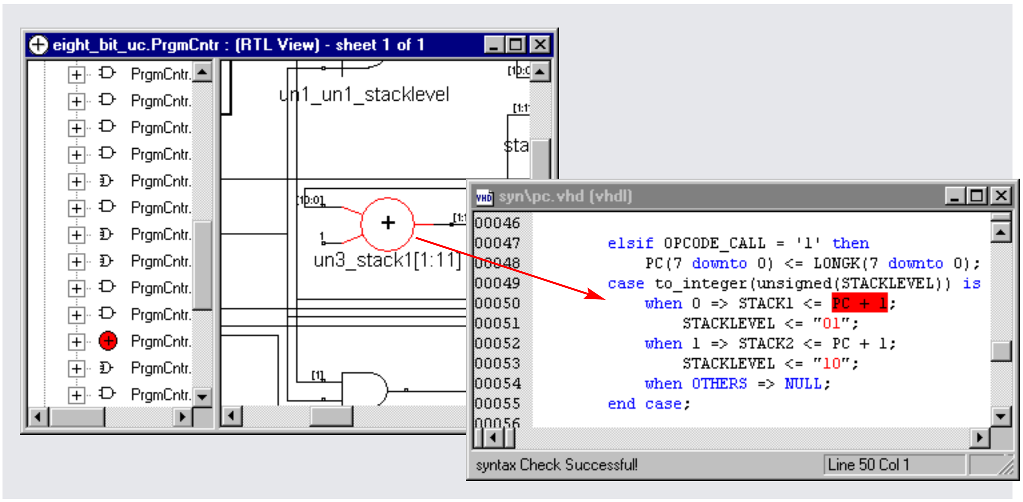
Un-Highlight Selection (*?):

[✓] Jump to location

add

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.

3. Crossprobe from the schematic to the source code:
   - Double-click the incrementor symbol. The software displays the corresponding RTL code. The following figure shows VHDL source code; if you are using the Verilog files, the source code will look different.



```
elsif OPCODE_CALL = '1' then
    PC(7 downto 0) <= LONGK(7 downto 0);
case to_integer(unsigned(STACKLEVEL)) is
    when 0 => STACK1 <= PC + 1;
        STACKLEVEL <= "01";
    when 1 => STACK2 <= PC + 1;
        STACKLEVEL <= "10";
    when OTHERS => NULL;
end case;
```

syntax Check Successful! Line 50 Col 1

   - Close the source file window and use Push/Pop mode to return to the top level.

4. Crossprobe from the source code.
   - In the RTL view, push down into the ALU block.
   - In the Project view, double-click `alu.vhd` to open the source code file.
   - From the text file window, press Ctrl-f and type longq <= as the search criteria. Click Find Next.
   - Highlight the entire line (line 60) that begins with longq <=. The corresponding logic is highlighted in the RTL view.
   - Click Cancel to close the Find dialog box. Close the Text Editor window.

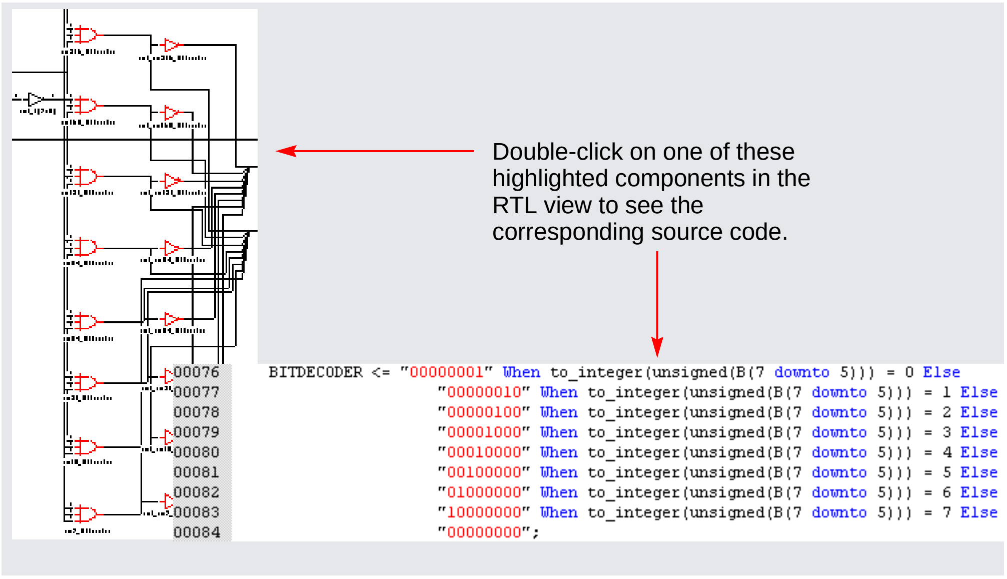
5. Find a bitdecoder and crossprobe. From the RTL view, press Ctrl-f or select Edit->Find and do the following:
   - Select the Instances tab and set the search range to Entire Design.
   - In the Highlight Search (*?) field, type *deco*, and click Find 200. The Unhighlighted selection list is now shorter, and only lists instances that

match the search criteria. For details about using wildcards, see the *Results Analysis* chapter of the *Synplify User Guide*.

– Click All-> to move the entire list to the Highlighted box. The schematic highlights the bitdecoder instances in red.

– Click OK to close the Object Query dialog box.

– With the instances selected in the RTL view, put your cursor over one of the selected instances and double-click. The corresponding source code for the bitdecoder definition in the `alu.vhd` file opens. Verilog source code will be different.


Double-click on one of these highlighted components in the RTL view to see the corresponding source code.

```
BITDECODER <= "00000001" When to_integer(unsigned(B(7 downto 5))) = 0 Else
              "00000010" When to_integer(unsigned(B(7 downto 5))) = 1 Else
              "00000100" When to_integer(unsigned(B(7 downto 5))) = 2 Else
              "00001000" When to_integer(unsigned(B(7 downto 5))) = 3 Else
              "00010000" When to_integer(unsigned(B(7 downto 5))) = 4 Else
              "00100000" When to_integer(unsigned(B(7 downto 5))) = 5 Else
              "01000000" When to_integer(unsigned(B(7 downto 5))) = 6 Else
              "10000000" When to_integer(unsigned(B(7 downto 5))) = 7 Else
              "00000000";
```

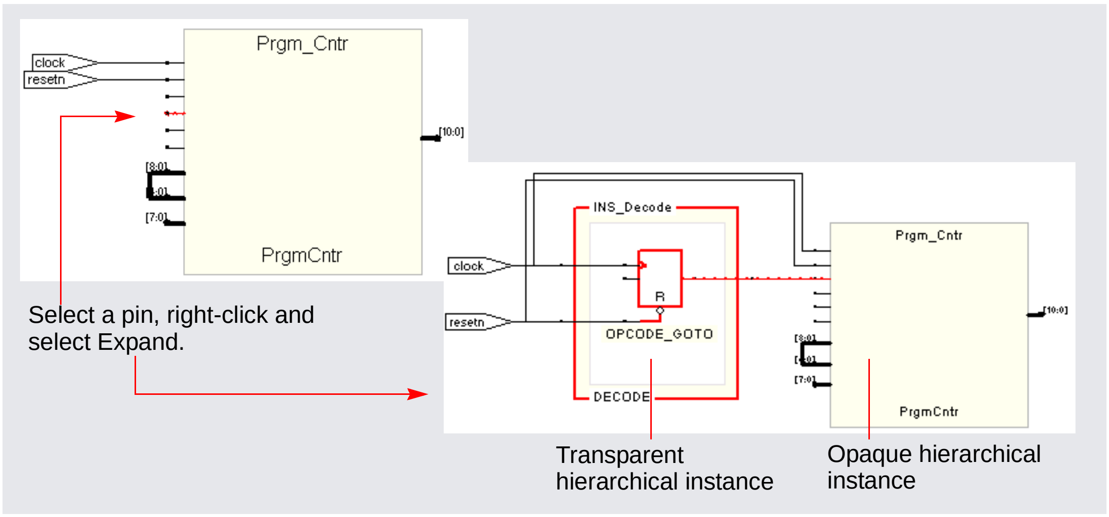
To crossprobe from the bitdecoder definition to the schematic, you must have an open RTL window, and you must select all of the bits that make up the bitdecoder definition in the source code.

6. 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 *Examine the RTL View and Navigate Hierarchy* on page 11), 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 *Results Analysis* chapter of the *Synplify User Guide*.

1. In the top-level RTL view, filter and expand pin connections:
   - Select Prgm_Cntr and press F12 or select , the Filter icon. The schematic is filtered, and only the selected object is displayed.
   - Select the pin shown in the following figure, right-click, and select Expand. The software expands the connection to the next register and displays it. This register is inside Ins_decode, so the software indicates hierarchy with a transparent hierarchical instance (a hollow bounding box surrounding the lower-level logic connection).

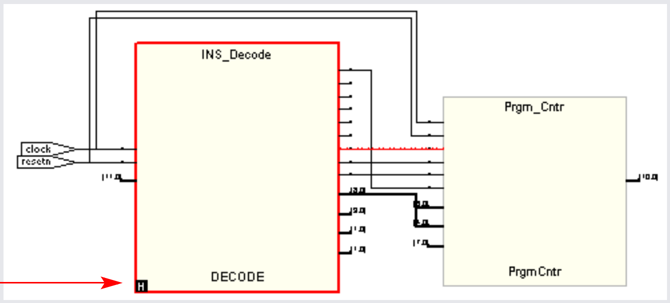

clock
resetn
Prgm_Cntr
[10:0]
[8:0]
[7:0]
PrgmCntr
Select a pin, right-click and select Expand.
INS_Decode
R
OPCODE_GOTO
DECODE
Transparent hierarchical instance
Opaque hierarchical instance

   - Click Back until you return to the top level.

2. In the top-level RTL view, do the following:
   - Select INS_Decode, right-click and select Hide Instances. You see a small H in the lower left corner of the instance, which indicates all lower-level hierarchy is “hidden” from certain operations, like expanding.
   - Click in a blank area so that nothing is selected.

– Repeat the instructions in the previous step and expand the same pin. The results are different, because the internal hierarchy of the hidden instance is not expanded.

Internal hierarchy of hidden instance is not expanded.



– Click Back until you return to the top-level view.

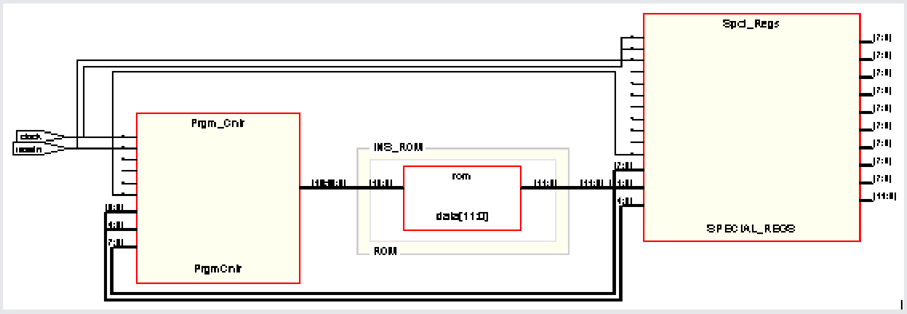
– Click the RTL icon and open another window with the top-level RTL view. Zoom in and look at the lower left of INS_Decode. It is not hidden in this window, although it is hidden in the first RTL window. You can hide different portions of the design hierarchy in different windows. Close the second RTL window.

– Return to the first RTL window and select INS_Decode. Right-click and select Unhide Instances. The instance is no longer hidden.

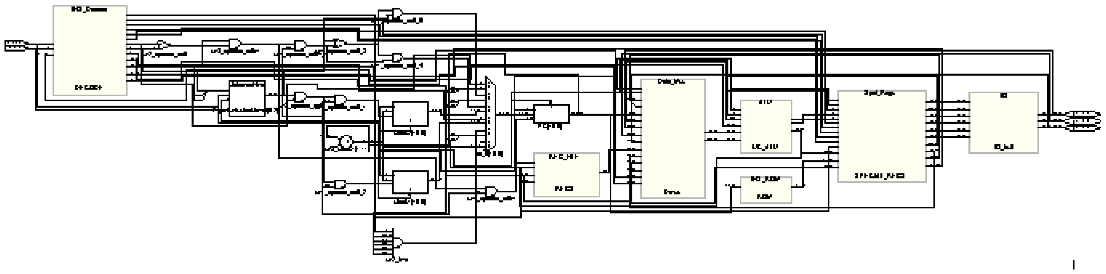
3. View the connections between selected instances.

– In the top-level view, click Prgm_Cntr. Press Ctrl and 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.

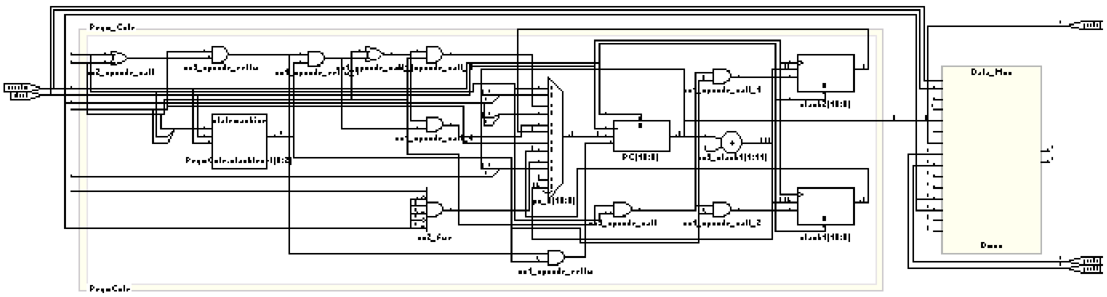


4. Push into a transparent instance. You cannot push into a transparent instance like INS_ROM from any point within the hollow bounding box.
   - Select INS_ROM, and click the Push/Pop Hierarchy icon. To push into INS_ROM, put the cursor over the instance border and click when you see the cursor change to an arrow pointing downwards.
   - Push into the ROM block. You see a text file with the ROM data table. Close the text window. Exit Push/Pop mode and return to the top level.
5. Flatten hierarchy.
   - In the top-level view, select Flatten Schematic. The software flattens the entire design. Return to the top level by right-clicking and selecting Unflatten Schematic. You cannot use the Back button, because this is a flattened view, not a filtered view. In a flattened view, there is no history.
   - In the 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 top-level 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 and Data_Mux. Click the Filter 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 Back as needed to 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, but do not close it.

The rest of the tutorial varies slightly, depending on the technology. To work with an Altera technology, go to *Altera Flow* on page 22 for the next step; for a Xilinx technology, go to *Xilinx Flow* on page 33 for the next step. If you do not use either of these vendors, you can follow the methodology used in either of these flows and substitute device options specific to your vendor.

Altera Flow

The Altera design flow in this tutorial uses the APEX II technology. The following sections show you how to

- Set Altera Constraints, next
- Set Altera Device Options on page 24
- Run Synthesis on page 26
- Analyze the Synthesis Results on page 27
- Rerun Synthesis on page 31

To work through the tutorial with the Xilinx design flow, see *Xilinx Flow* on page 33. If you do not use Altera or Xilinx, you can still follow along with the tutorial, using device options specific to your vendor.

Set Altera Constraints

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

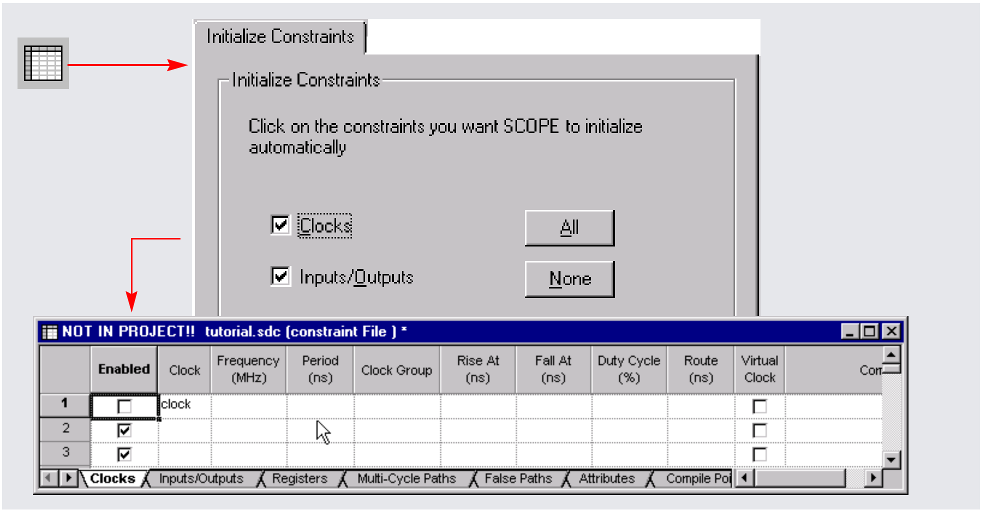
You can set constraints in a text file that you can create with any text editor, but it is easier to use the SCOPE (Synthesis Constraint Optimization Environment) interface. The SCOPE interface consists of a spreadsheet where you can enter constraints.

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:
   – Clicking the Constraint file (SCOPE) icon in the toolbar. (▦)
   – Selecting File - >New, choosing Constraint file (SCOPE) in the dialog box, and clicking OK.

2. Click OK in the Create a New Constraint File dialog box.

   The SCOPE window opens, with the most common constraints, clock frequency, and input/output delays initialized.



Initialize Constraints

Initialize Constraints

Click on the constraints you want SCOPE to initialize automatically

☑ Clocks

☑ Inputs/Outputs

All

None

NOT IN PROJECT!! tutorial.sdc (constraint File) *

	Enabled	Clock	Frequency (MHz)	Period (ns)	Clock Group	Rise At (ns)	Fall At (ns)	Duty Cycle (%)	Route (ns)	Virtual Clock	Com
1	☐	clock								☐	
2	☑									☐	
3	☑									☐	

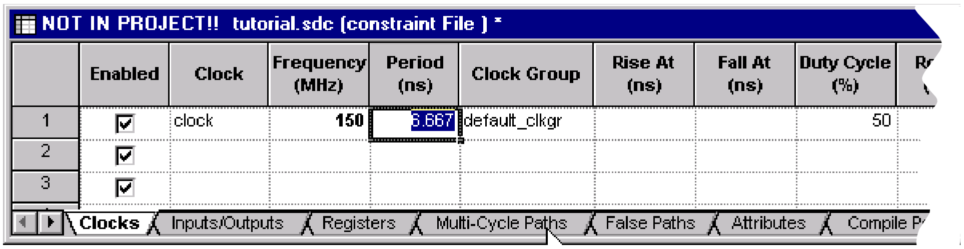
Clocks / Inputs/Outputs / Registers / Multi-Cycle Paths / False Paths / Attributes / Compile Poi

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.

   – Click the check box in the Enabled column to enable the clock constraint.

   – Enter 150 in the Frequency column to set the clock frequency, and press Enter. This design has only one clock, so setting the clock frequency is the same as setting a global frequency. When you press Enter, the value in the Frequency column automatically sets the period. The clock is assigned to the default clock group.



NOT IN PROJECT!! tutorial.sdc (constraint File) *

	Enabled	Clock	Frequency (MHz)	Period (ns)	Clock Group	Rise At (ns)	Fall At (ns)	Duty Cycle (%)	R
1	☑	clock	150	6.667	default_clkgr			50	
2	☑								
3	☑								

Clocks / Inputs/Outputs / Registers / Multi-Cycle Paths / False Paths / Attributes / Compile P

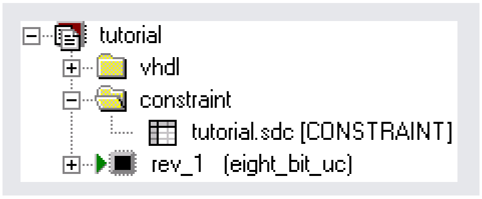
4. Click the ( ) icon, or select File -> Save and save the file as `tutorial.sdc`. Close the SCOPE window.
5. Click Yes in the dialog box that asks you if you want to add the file to the project.

   You should now have the following files in the project:

   – A vhdl folder that contains the source files

   – A constraint folder with the constraint file (tutorial.sdc)

   – An implementation folder (rev_1)



tutorial
vhdl
constraint
tutorial.sdc [CONSTRAINT]
rev_1 (eight_bit_uc)

Set Altera Device Options

At this point, you set the device options for the first implementation (rev_1).

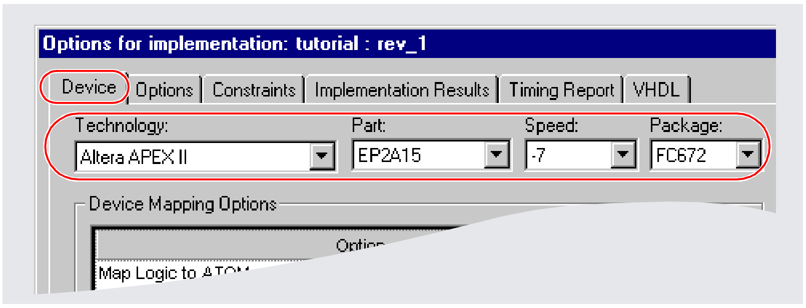
1. Select rev_1, and either click the Target option just above the Run button or select Project -> Implementation Options.

   The dialog box that opens lists the implementation (rev_1) at the top. It has many tabs, and opens with the Device tab (the technology parameters) displayed.
2. In the Device section,

   – Set Technology to Altera APEXII. The software supports a large selection of target technologies, with specific algorithms for optimizing each technology for best results.

   – Leave the other defaults: Part EP2A15, Speed -7, and Package FC672.

– Do not change any other device options.


Options for implementation: tutorial : rev_1

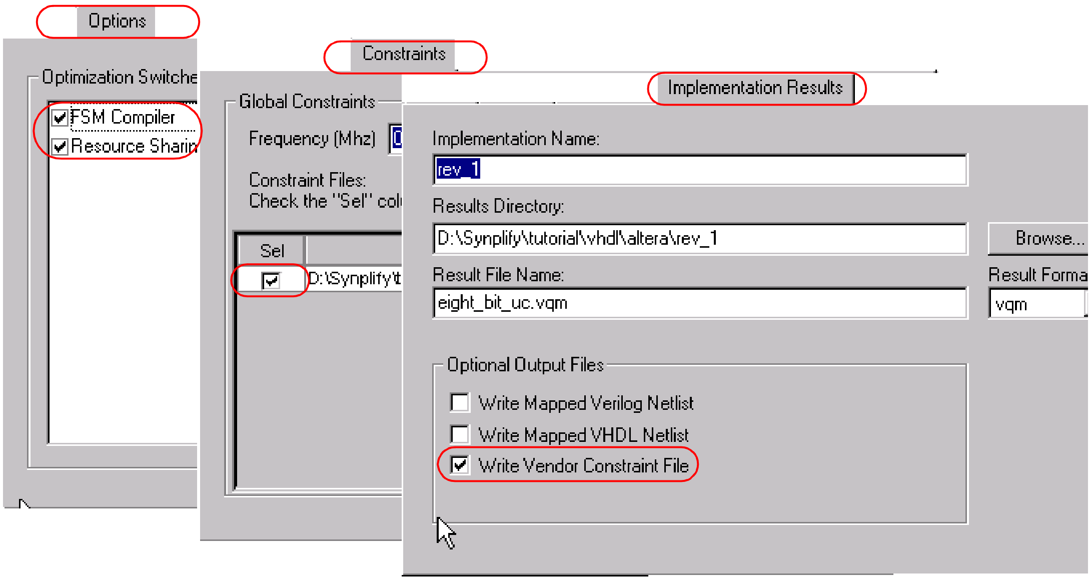
Technology: Altera APEX II
Part: EP2A15
Speed: -7
Package: FC672

3. Check the following on the next few tabs:

 – On the Options tab, leave the default settings for the optimization switches (Symbolic FSM Compiler and Resource Sharing on).

 – On the Constraints tab, make sure the constraint file is checked.

 – On the Implementation Results tab, enable Write Vendor Constraint File.



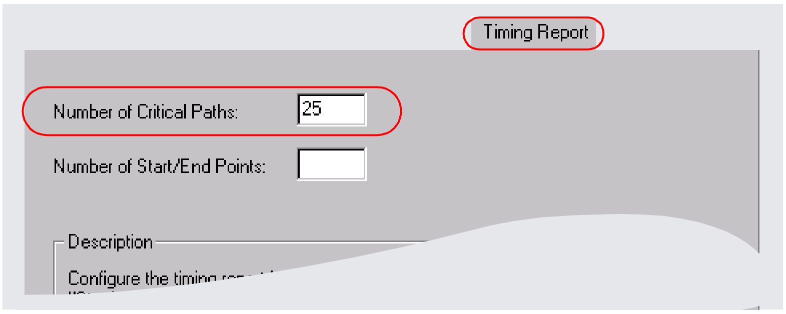
[✓] FSM Compiler
[✓] Resource Sharin

[✓] D:\Synplify\t

Implementation Name: rev_1
Results Directory: D:\Synplify\tutorial\vhdl\altera\rev_1
Result File Name: eight_bit_uc.vqm
Result Forma: vqm

Optional Output Files
[ ] Write Mapped Verilog Netlist
[ ] Write Mapped VHDL Netlist
[✓] Write Vendor Constraint File

4. Click the Timing Report tab, and set Number of Critical Paths to 25. This option determines the number of critical paths reported in the timing report generated after synthesis.



Number of Critical Paths: 25
Number of Start/End Points:

5. Do not make any changes on the VHDL tab. Click OK.

   You have already specified the top-level module in the project window by putting it last in the list. If you had not done that, you would specify the top-level module in the VHDL tab.

Run Synthesis

Once you have set up your files, synthesis is simple.

1. Make sure you have the prerequisites for synthesis:
   – Source files
   – Target technology (device options)
   – An optional constraint file
2. Click the Run button to start synthesis.

   The software goes through two synthesis phases, compilation and mapping, and it reflects these stages in red letters in the lower right of the project window. 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. The Implementation Results view lists the files that are generated as a result of synthesis.

When synthesis is complete, the software displays this message:

Done !

Analyze the 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 29
- Analyze Critical Paths in the Technology View on page 30

Examine the Technology View

Use the Technology view to graphically check the synthesis results.

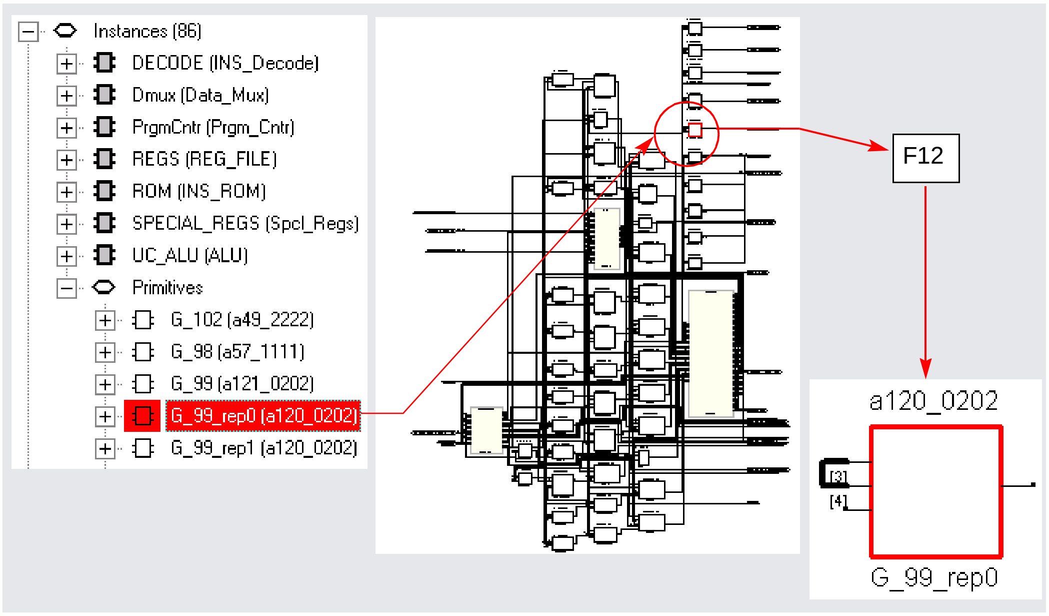
1. To see the graphical results of your run, click the ( ) icon to open the Technology view.

   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 the 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.

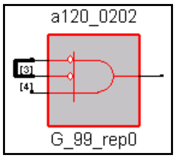
– To reduce clutter in the schematic, select Options->Schematic Options and deselect Show Cell Interior. Click OK.

– In the Hierarchy Browser on the left side of the Technology view, click the plus sign next to Instances, and then select an instance. The component is highlighted in red on the schematic. If you have multiple sheets (small sheet size is a preference set with Options->Schematic Options), the Technology view automatically moves to the correct sheet with the selected component.



– 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 more details, select Options->Schematic Options and enable Show Cell Interior and Show Pin Name. Click OK. You see the interior of the cell. Roll the cursor over the pins to see the pin names. You also 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 *Examine the RTL View and Navigate Hierarchy* on page 11, *Find and Crossprobe* on page 15, and *Filter, Expand, Hide, and Dissolve* on page 18 to examine how the design was implemented for this technology.
- Return to the top level by clicking Back. If you are on another sheet of the schematic, you can alternatively return to the top level by pressing Ctrl-g or holding down the right mouse button in the Technology view and selecting View Sheets from the popup menu. Then select Sheet 1 from the list, and click OK.
- Close this window.

Check Timing

You can check timing results in the log file. .

1. Open the log file (eight_bit_uc.srr) by selecting the View Log button in the Project window.

   You see a window with the log file.

2. In the log file window, scroll down to the Performance Summary section.

   You see details of the clock information. This design does not meet timing because it has a negative slack time. Positive or 0 slack times indicate that you have met or are under the timing constraint.

   If your critical path is not exactly the same because of ongoing software optimizations, apply the general techniques described to your design.

3. Scroll a little further (Worst Paths Information) to see the detailed critical path. A table shows all the points on the critical path after this summary:

```
Path information for path number 1:
 - Setup time:5.500
 = Required time:1.167

 - Propagation time:1.352
 = Slack (critical):-0.185

 Starting point:DECODE.FWE / regout
 Ending point:REGS.mem.I_1 / wren
```

Close the log file window. You can now check the critical path in the Technology view.

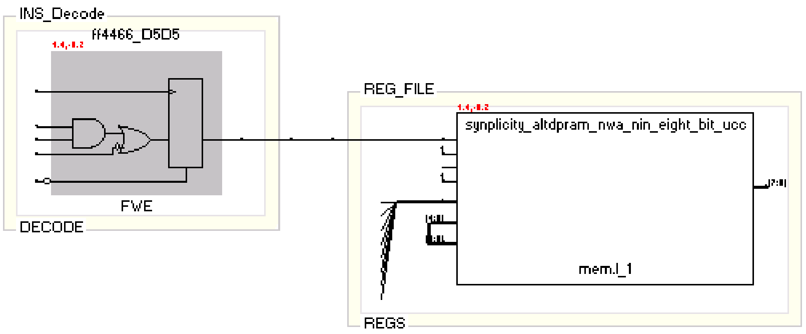
Analyze Critical Paths in the Technology View

To analyze your critical path, follow this procedure here. There is no need to flatten the hierarchical schematic.

1. With an open Technology view window, select the Critical Path icon (⏰), or right-click in the Technology view window and select Show Critical Path.

   The Technology view graphically displays the path you saw in the log file. Your critical path might differ from the one shown, if you are not using the version of the software for which the tutorial was written. 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. The red numbers contain timing information: the first is the delay, and the second, the slack time.

   If you do not see the timing information displayed, enable HDL Analyst->Show Timing Information.



2. Zoom into the timing information. You can see that the slack (second number) is negative, which means that you did not meet timing.

3. You can now use other techniques to analyze your path and design further. This is what you would normally do in a design, but this step is optional in this tutorial.

   - 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 18.
   - To return to the critical path view, click Back or click the Critical Path icon. If Back is inactive (you flattened the path), click the Critical Path icon to return to the critical path view.

   For the purposes of this tutorial, we will handle this critical path by adding a two-cycle path constraint and resynthesizing the design. See the *Synplify Reference Manual* for details about other constraints and attributes you can add.

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

Rerun Synthesis

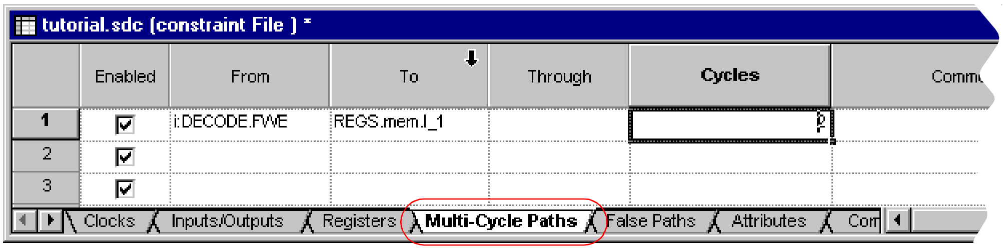
This section guides you through the post-analysis phase, where you set constraints, rerun synthesis, and check your results.

Resynthesize the Design

For the purposes of this tutorial, you now add a timing constraint to the critical path and resynthesize the design.

1. Make sure you have the filtered view of the critical path open.
2. Open the constraints file.
   - Double-click the constraint file (`tutorial.sdc`) in the Project view to open the file. The SCOPE window opens.
   - Select the Multi-cycle Paths tab.
3. Add a constraint from the start point to the end point using these steps:

- Check the detailed critical path section of the log file for the start and end points. To find the section in the log file, use Ctrl-f and search for crit or Worst Paths.
- Click in the SCOPE From column, and select i:DECODE.FWE as the start point. If your critical path is not the same as the one shown, specify the starting point of your critical path in the From column.
- Hold down the mouse button over Regs.mem.l_1 in the Technology view, drag it to the SCOPE To column, and release the button. If your critical path is different, specify the end point in the To column.
- Set Cycles to 2 and check the Enabled column to apply the constraint.


tutorial.sdc (constraint File) *

	Enabled	From	To	Through	Cycles	Comm
1	☑	i:DECODE.FWE	REGS.mem.l_1		2	
2	☑					
3	☑					

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

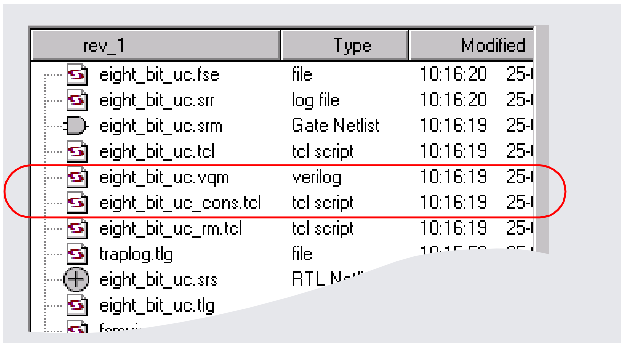
Check the Results

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

1. Check the results in the log file as described previously in *Check Timing* on page 29.

   The design now meets the timing requirements. There is a new critical path listed.

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



rev_1	Type	Modified
eight_bit_uc.fse	file	10:16:20 25-
eight_bit_uc.srr	log file	10:16:20 25-
eight_bit_uc.srm	Gate Netlist	10:16:19 25-
eight_bit_uc.tcl	tcl script	10:16:19 25-
eight_bit_uc.vqm	verilog	10:16:19 25-
eight_bit_uc_cons.tcl	tcl script	10:16:19 25-
eight_bit_uc_rm.tcl	tcl script	10:16:19 25-
traplog.tlg	file	
eight_bit_uc.srs	RTL N	
eight_bit_uc.tlg		

The software generates vendor-specific netlists with the attributes and constraints carried forward. This ensures that the design is optimized for the target technology. The `eight_bit_uc.vqm` file is the netlist for the place-and-route tools, and the `eight_bit_uc_cons.tcl` 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 the design. You can use the Synplify interface to crossprobe and debug your designs further, or use the synthesis output files to place and route.

Xilinx Flow

The Xilinx flow uses Virtex technology. The following sections discuss these topics:

- Set Xilinx Constraints, next
- Set Xilinx Device Options on page 36
- Run Synthesis on page 38
- Analyze the Synthesis Results on page 39
- Rerun Synthesis on page 43

To work through the tutorial with the Altera design flow, see *Altera Flow* on page 22. If you do not use Altera or Xilinx, you can follow along with the tutorial using device options specific to your vendor.

Set Xilinx Constraints

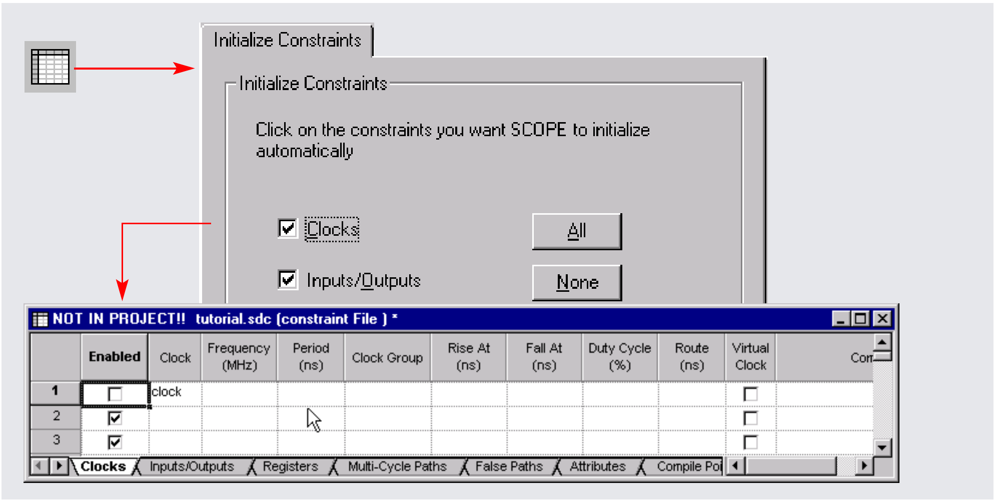
Design constraints are optional, but most designers use them to define the frequency goals and describe the environment around the design. For designs without aggressive timing goals, you can just set the clock frequency.

You can set constraints in a text file (see the *Synplify Reference Manual*), but it is easier to use the SCOPE (Synthesis Constraint Optimization Environment) interface. This tool provides an easy-to-use spreadsheet interface for entering constraints.

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

1. Start the SCOPE tool in the open project window by doing one of the following:
   - Click the Constraint file (SCOPE) icon ( ) in the toolbar.
   - Select File ->New, choose Constraint file (SCOPE) in the dialog box, and click OK.
2. Leave the defaults selected, and click OK.

   The SCOPE window opens, with the most common constraints, frequency, and I/O delays initialized. The window consists of a spreadsheet interface with tabs for different kinds of constraints. The Compile Points tab is technology-dependent, so your design might not include it.



Initialize Constraints

Click on the constraints you want SCOPE to initialize automatically

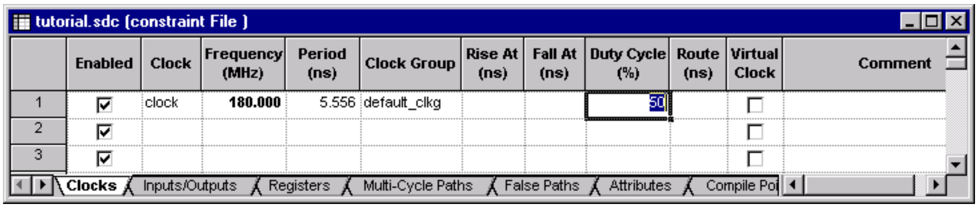
[✓] Clocks

[✓] Inputs/Outputs

NOT IN PROJECT!! tutorial.sdc (constraint File) *

	Enabled	Clock	Frequency (MHz)	Period (ns)	Clock Group	Rise At (ns)	Fall At (ns)	Duty Cycle (%)	Route (ns)	Virtual Clock	Com
1	☐	clock								☐	
2	✓									☐	
3	✓									☐	

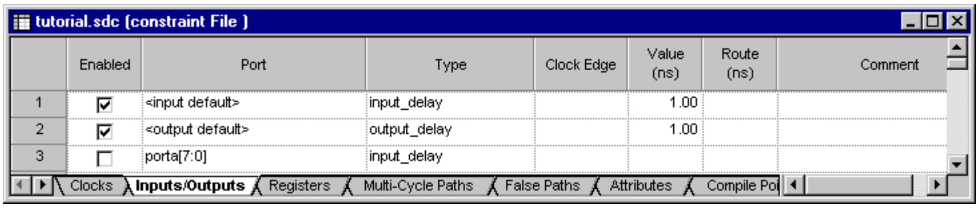
3. Set a clock frequency constraint as follows:
   - Select the Clocks tab at the bottom of the SCOPE window, if it is not already selected.
   - Enable the clock constraint by clicking in the Enabled column.
   - Enter 180 in the Frequency column to set the clock frequency and press Return. This design has only one clock, so setting the clock frequency is the same as setting a global frequency. When you press Return, the value in the Frequency column automatically sets the period. The clock is assigned to the default clock group.



tutorial.sdc (constraint File)

	Enabled	Clock	Frequency (MHz)	Period (ns)	Clock Group	Rise At (ns)	Fall At (ns)	Duty Cycle (%)	Route (ns)	Virtual Clock	Comment
1	☑	clock	180.000	5.556	default_clkg			50		☐	
2	☑									☐	
3	☑									☐	

4. To set input/output delay constraints,
   - Click the Inputs/Outputs tab.
   - Enable <input default> and <output default> by clicking the corresponding boxes in the Enabled column.
   - Set a value of 1 ns for each of them.



tutorial.sdc (constraint File)

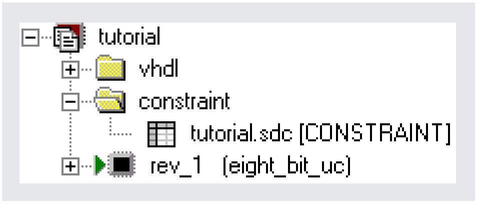
	Enabled	Port	Type	Clock Edge	Value (ns)	Route (ns)	Comment
1	☑	<input default>	input_delay		1.00		
2	☑	<output default>	output_delay		1.00		
3	☐	porta[7:0]	input_delay				

5. Save the constraint file by clicking the ( ) icon, or by selecting File - Save. Save it as `tutorial.sdc`.

6. Select Yes in the dialog box that asks you if you want to add the file to the project. Close the SCOPE window.

   The software saves it as `tutorial.sdc` and puts it in a folder in the Project view. You should now have the following files in the project:
   - A vhdl folder that contains the source files

- A constraint folder with the constraint file
- An implementation folder (rev_1)



```
tutorial
  vhdl
  constraint
    tutorial.sdc [CONSTRAINT]
  rev_1   (eight_bit_uc)
```

Set Xilinx Device Options

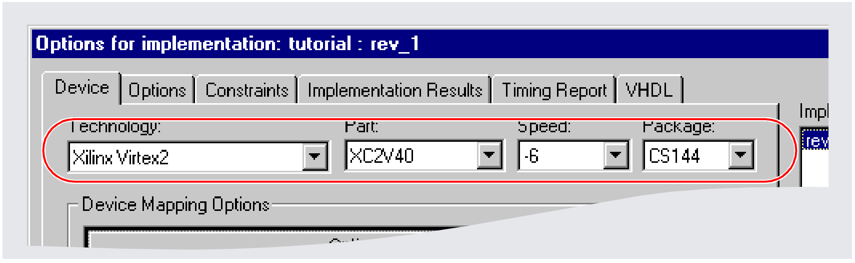
At this point, you set the device options for the first implementation.

1. Either click the Target options above the Run button or select Project -> Implementation Options.

   The dialog box that opens lists the implementation (rev_1) at the top. It has many tabs, and opens with the Device tab (the technology parameters) displayed.

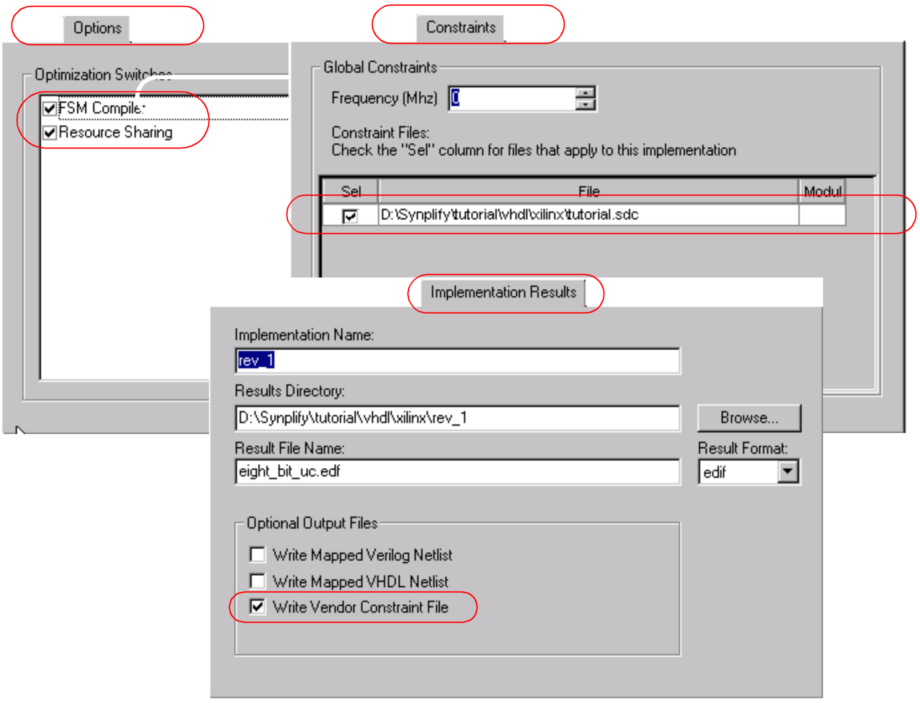
2. In the Device section,

   - Set Technology to Xilinx Virtex2. The software supports a large selection of target technologies. The software has specific algorithms for optimizing each technology for best results. For example, there are special fanout optimizations for Xilinx Virtex designs.
   - Leave the other defaults: Part XC2V40, Speed -6, and Package CS144.
   - Do not change any other device mapping options.



Technology: Xilinx Virtex2
Part: XC2V40
Speed: -6
Package: CS144

3. On the other tabs, check the following:
   - On the Options tab, leave the default optimization switches (Symbolic FSM Compiler and Resource Sharing) enabled.
   - On the Constraints tab, make sure the constraint file (`tutorial.sdc`) is checked.
   - On the Implementation Results tab, make sure the Write Vendor Constraint File option is checked.



[✓] FSM Compiler
[✓] Resource Sharing

Frequency (Mhz) 0

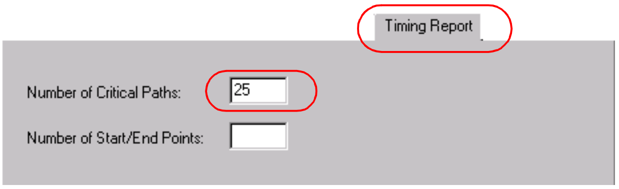
Constraint Files:
Check the "Sel" column for files that apply to this implementation

Sel	File	Modul
✓	D:\Synplify\tutorial\vhdl\xilinx\tutorial.sdc	

Implementation Name: rev_1
Results Directory: D:\Synplify\tutorial\vhdl\xilinx\rev_1
Result File Name: eight_bit_uc.edf
Result Format: edif

[ ] Write Mapped Verilog Netlist
[ ] Write Mapped VHDL Netlist
[✓] Write Vendor Constraint File

4. Click the Timing Report tab, and set Number of Critical Paths to 25. This option determines the number of critical paths reported in the timing report generated after synthesis.



Number of Critical Paths: 25
Number of Start/End Points:

5. Do not make any changes on the VHDL tab. Click OK.

   You have already specified the top-level module in the project window by putting it last in the list. If you had not done that, you can specify the top-level module in the VHDL tab. You are now ready to set some design constraints.

Run Synthesis

You can now run synthesis.

1. Make sure you have the prerequisites for synthesis:
   – Source files
   – Target technology (device options)
   – An optional constraint file
2. Click the Run button to start synthesis.

   The software goes through two synthesis phases, compilation and mapping, and it reflects these stages in red letters in the lower right of the project window. Compilation is the creation of a technology-independent boolean structure, and mapping is the technology-specific 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. The Implementation Results view lists the files that are generated as a result of synthesis.

   When synthesis is complete, the software displays this message:

Analyze the Synthesis Results

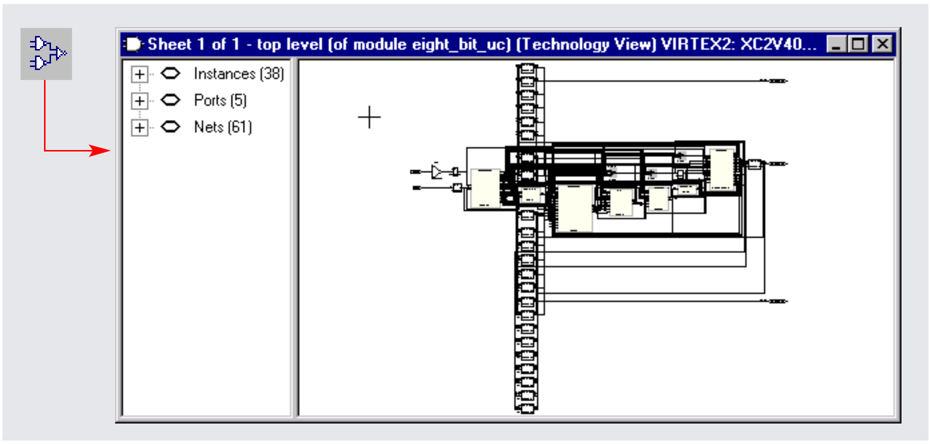
After you have run synthesis, you analyze the results. This section describes how to do the following:

- Examine the Technology View, next
- Check Timing on page 41
- Analyze Critical Paths in the Technology View on page 42

Examine the Technology View

Use the Technology view to graphically check the synthesis results.

1. To see the graphical results of your run, click the Technology view icon ( ) to open the Technology view.

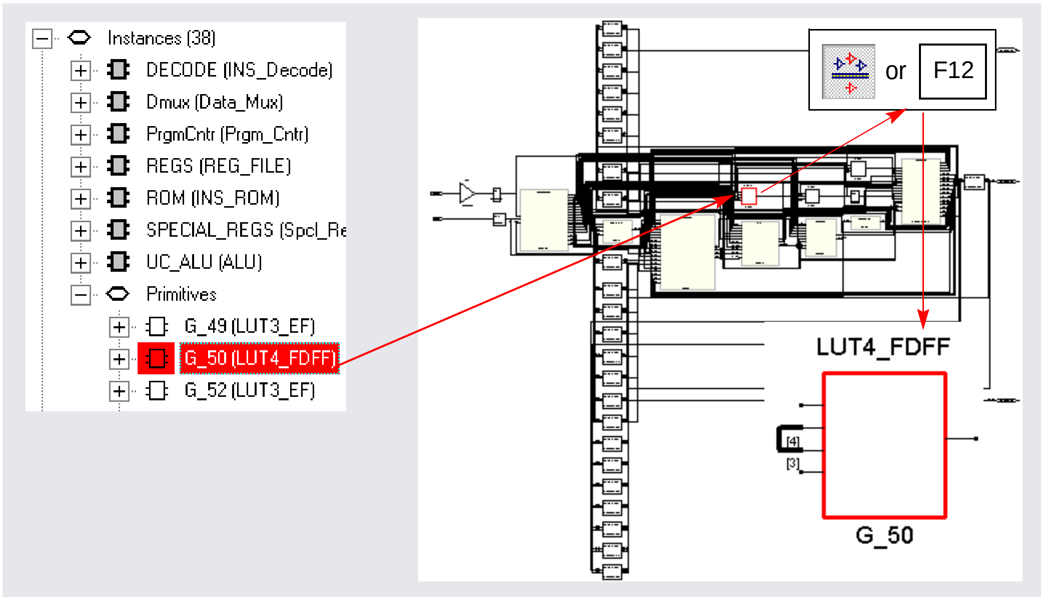

Sheet 1 of 1 - top level (of module eight_bit_uc) (Technology View) VIRTEX2: XC2V40...
Instances (38)
Ports (5)
Nets (61)

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 instances as follows. If you are not using the version of the software for which the tutorial was written, your design may be implemented with different components, because of ongoing optimizations to the technology and the software.

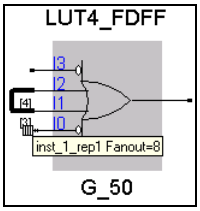
– To reduce clutter in the schematic, select Options->Schematic Options and deselect Show Cell Interior. Click OK.

– In the Hierarchy Browser on the left side of the Technology view, expand Instances and then Primitives, and click an instance. This example shows a LUT selected. The component is highlighted in red on the schematic. If you have multiple sheets (small sheet size is a preference set with Options->Schematic Options), the Technology view automatically moves to the correct sheet with the selected component.



– 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 more details, select Options->Schematic Options and enable Show Cell Interior and Show Pin Name. Click OK. You see the interior of the cell. Roll the cursor over the pins to see the pin names. You also 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 *Examine the RTL View and Navigate Hierarchy* on page 11, *Find and Crossprobe* on page 15, and *Filter, Expand, Hide, and Dissolve* on page 18 to examine how the design was implemented for this technology.
- Return to the top level by clicking Back. If you are on another sheet of the schematic, you can alternatively return to the top level by pressing Ctrl-g or holding down the right mouse button in the Technology view and selecting View Sheets from the popup menu. Then select Sheet 1 from the list, and click OK.
- Close this window.

Check Timing

You can check timing results in the log file. .

1. Open the log file (eight_bit_uc.srr) by selecting the View Log button in the Project window.

   You see a window with the log file.

2. In the log file window, scroll down to the Performance Summary section.

   You see details of the clock information. The details may vary, because of ongoing optimizations to the software. This design does not meet timing because it has a negative slack time. Positive or 0 slack times indicate that you have met or are under the timing constraint.

3. Scroll a little further (or you can search for Worst Paths Information) to see the detailed critical path. A table shows all the points on the critical path after this summary:

```
Path information for path number 1:
- Setup time: -0.432
= Required time: 5.988

- Propagation time:6.450
= Slack (critical) :-0.462

Starting point: Dmux.ALUA[2] / Q
Ending point:UC_ALU.ALUZ / D
```

4. Close the log file window.

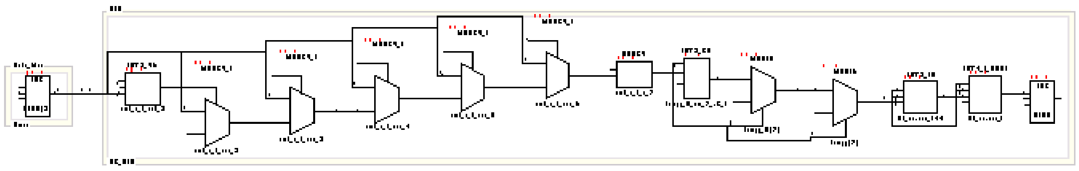
   You can now check the critical path in the Technology view.

Analyze Critical Paths in the Technology View

You do not need to flatten the design to view the critical path. You can view the hierarchical critical path.

1. With an open Technology view window, select the Critical Path icon (⏱), or right-click in the Technology view and select Show Critical Path.

   The Technology view graphically displays the path you saw in the log file. Your critical path might differ from the one shown. The critical path view is a filtered view with just the instances on the critical path. The following figure shows the critical path, with transparent instances to indicate the design hierarchy. The red numbers specify timing information; the first is the delay, and the second the slack time.



   If you do not see the kind of critical path information shown in the figure, enable HDL Analyst->Show Timing Information.

2. Zoom into the timing values. You can see that the slack (second number) is negative, which means that you did not meet timing.

3. You can now use other techniques to analyze your path and design further. This is what you would normally do in a design, but this step is optional in this tutorial.
   - Check the corresponding RTL code, by double-clicking an object in the Technology view.
   - Filter and expand paths of the path, using the techniques described in *Filter, Expand, Hide, and Dissolve* on page 18.
   - To return to the critical path view, click Back or click the Critical Path icon. If Back is inactive (you flattened the path), click the Critical Path icon to return to the critical path view.

   For the purposes of this tutorial, we will handle this critical path by adding a two-cycle path constraint and resynthesizing the design. See the *Synplify Reference Manual* for details about other constraints and attributes you can add.

4. Leave the filtered critical path view open, and close any other open windows.

Rerun Synthesis

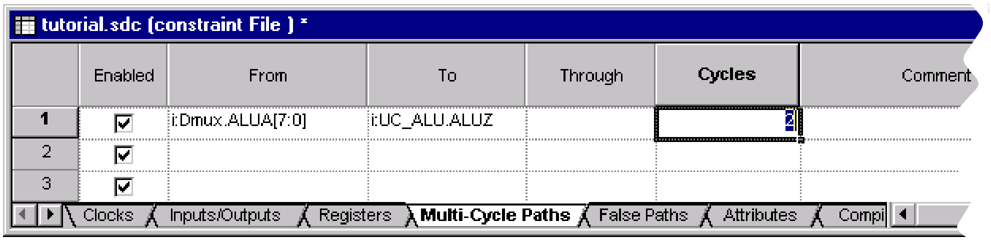
To meet timing, you set multicycle constraints on the start and end points of the critical path. You add the constraints and resynthesize the design.

Resynthesize the Design

Set new constraints and resynthesize your design.

1. Make sure you have the filtered view of the critical path open.
2. Open the constraints file.
   - Double-click on the constraint file (`tutorial.sdc`) in the Project view to open the file. The SCOPE window opens.
   - Select the Multi-cycle Paths tab.

3. Add a constraint from the start point to the end point.
   - Check the detailed critical path section of the log file for the start and end points. To find the section in the log file, use Ctrl-f and search for crit or Worst Paths.
   - Click in the SCOPE From column, and select i:Dmux.ALUA[7:0] (the start point in the log file) from the pulldown menu. If your critical path is different, use the appropriate starting pint in the From column.
   - Click in the To column, and select i:UC_ALU.ALUZ (the end point) from the pulldown menu. If your critical path is different, use the appropriate starting pint in the To column.
   - Set Cycles to 2.
   - Ensure that the box in the Enabled column is checked, to apply the constraint.


tutorial.sdc (constraint File) *

	Enabled	From	To	Through	Cycles	Comment
1	✓	i:Dmux.ALUA[7:0]	i:UC_ALU.ALUZ		2	
2	✓					
3	✓					

You can also drag and drop objects from the schematic to the SCOPE window, instead of selecting them from the pull-down menu.

4. Save the constraint file, and minimize the SCOPE window.
5. Click the Run button to rerun synthesis. You can now check the results to see if the constraints eliminated the negative slack on the first path.

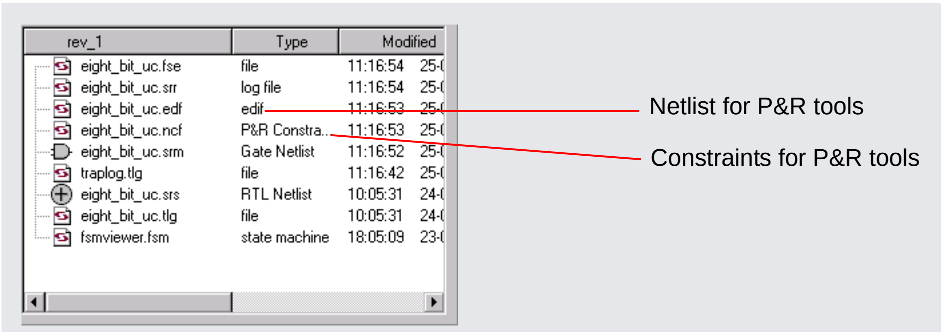
Check the Results

Check the results after applying the new constraints.

1. Check the results in the log file. For details, see *Check Timing* on page 41.

   The log file shows the next most critical path. The negative slack on the first path has been eliminated.

2. Close the different windows you have open.
3. Check the output files in the Implementation Results view.



rev_1	Type	Modified	
eight_bit_uc.fse	file	11:16:54	25-
eight_bit_uc.srr	log file	11:16:54	25-
eight_bit_uc.edf	edif	11:16:53	25-
eight_bit_uc.ncf	P&R Constra...	11:16:53	25-
eight_bit_uc.srm	Gate Netlist	11:16:52	25-
traplog.tlg	file	11:16:42	25-
eight_bit_uc.srs	RTL Netlist	10:05:31	24-
eight_bit_uc.tlg	file	10:05:31	24-
fsmviewer.fsm	state machine	18:05:09	23-

The software generates vendor-specific netlists with the attributes and constraints carried forward. This ensures that the design is optimized for the target technology.

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

Index