Foundation Series ISE 3.3i Quick Start Guide

About This Manual

This guide should be used as the initial learning tool for designers who are unfamiliar with the features of the Foundation Series Integrated Synthesis Environment (ISE) 3.x software.

Note This Xilinx software release is certified as Year 2000 compliant.

Manual Contents

This guide covers the following topics.

- Chapter 1, “Introduction” describes the key features of the ISE software and lists available documentation.
- Chapter 2, “Setting Up the Tools,” gives instructions for installing the ISE software and provides you with information about the type of computer you need to successfully implement your designs.
- Chapter 3, “Software Overview,” describes the capability and flexibility of the ISE software.
- Chapter 4, “Basic Tutorial” provides a step-by-step example explaining how to use the basic ISE tools.
- The “Glossary,” defines some of the commonly used terms in this Guide.

Additional Resources

For additional information, go to `http://support.xilinx.com`. The following table lists some of the resources you can access from this Web site. You can also directly access these resources using the provided URLs.

Resource	Description/URL
Tutorials	Tutorials covering Xilinx design flows, from design entry to verification and debugging `http://support.xilinx.com/support/techsup/tutorials/index.htm`
Answers Database	Current listing of solution records for the Xilinx software tools Search this database using the search function at `http://support.xilinx.com/support/searchtd.htm`
Application Notes	Descriptions of device-specific design techniques and approaches `http://support.xilinx.com/apps/appsweb.htm`
Data Book	Pages from *The Programmable Logic Data Book*, which contain device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging `http://support.xilinx.com/partinfo/databook.htm`
Xcell Journals	Quarterly journals for Xilinx programmable logic users `http://support.xilinx.com/xcell/xcell.htm`
Technical Tips	Latest news, design tips, and patch information for the Xilinx design environment `http://support.xilinx.com/support/techsup/journals/index.htm`

Conventions

This manual uses the following conventions. An example illustrates each convention.

Typographical

The following conventions are used for all documents.

- `Courier font` indicates messages, prompts, and program files that the system displays.

  ```
  speed grade: - 100
  ```

- **`Courier bold`** indicates literal commands that you enter in a syntactical statement. However, braces "{ }" in Courier bold are not literal and square brackets "[ ]" in Courier bold are literal only in the case of bus specifications, such as bus [7:0].

  ```
  rpt_del_net=
  ```

  `Courier bold` also indicates commands that you select from a menu.

  `File` → **`Open`**

- *Italic font* denotes the following items.

  - Variables in a syntax statement for which you must supply values

    `edif2ngd` *`design_name`*

  - References to other manuals

See the *Development System Reference Guide* for more information.

- Emphasis in text

  If a wire is drawn so that it overlaps the pin of a symbol, the two nets are *not* connected.

- Square brackets "[ ]" indicate an optional entry or parameter. However, in bus specifications, such as bus [7:0], they are required.

  edif2ngd *[option_name] design_name*

- Braces "{ }" enclose a list of items from which you must choose one or more.

  lowpwr ={**on**|**off**}

- A vertical bar " | " separates items in a list of choices.

  lowpwr ={**on**|**off**}

- A vertical ellipsis indicates repetitive material that has been omitted.

  ```
  IOB #1: Name = QOUT'
  IOB #2: Name = CLKIN'
  .
  .
  .
  ```

- A horizontal ellipsis "...." indicates that an item can be repeated one or more times.

  allow block *block_name loc1 loc2locn;*

Online Document

The following conventions are used for online documents.

- Red-underlined text indicates an interbook link, which is a cross-reference to another book. Click the red-underlined text to open the specified cross-reference.

- Blue-underlined text indicates an intrabook link, which is a cross-reference within a book. Click the blue-underlined text to open the specified cross-reference.

Contents

Chapter 3 Software Overview

Chapter 1

Introduction

This Quick Start Guide explains how to install the software, describes the basic software tools, and provides a basic tutorial.

This chapter contains the following sections.

- "Key Features"
- "Architecture Support"
- "Documentation"

Key Features

Following is a list of the key features supported for this release.

- Project Navigator

  The Project Navigator is the primary window interface used to access all of the Xilinx design tools.

- Design entry tools (Schematic Editor and HDL Editor)

  The HDL Editor supports Verilog, VHDL, and ABEL HDL.

  Xilinx provides the StateCAD Editor for designing State machines and HDL Bencher Editor for creating test benches.

- Design Synthesis

  The design synthesis tools create an EDIF netlist for HDL designs. Project Navigator supports three synthesis tools:

  - Xilinx Synthesis Tool (XST)
  - FPGA Express
  - Synplify / Synplify Pro (synthesis tool must be purchased separately)

- Design Implementation tools

  These tools place and route FPGAs or run the fitter for CPLDs. The design implementation tools also create back-annotated files for use in simulation.

- Design Constraints Graphical User Interfaces (GUIs)

  The Project Navigator contains three GUIs for creating constraints: the Xilinx Constraints Editor, Floorplanner, and the FPGA Express Constraints Editor.

- Third Party Tools

  - Model Technology Incorporated (MTI)

    The ISE software supports a variety of simulation tools produced by MTI. Xilinx provides the ModelSim Xilinx Edition (MXE) on a separate CD. Xilinx also supports the use of its software with the MTI PE and EE/SE simulation tools.

- Snapshots

  When you have successfully run your design through the design process, you can take a snapshot of the design. A snapshot saves all of the files that you have created which includes all of the files that were created during the design processing, that is, design entry, simulation, implementation, and programming.

  Note Snapshots are not compatible with the Design Manager Flow Engine. Projects created with Foundation ISE software cannot be opened within the Design Manager Flow Engine.

- Error Navigation to Solution Records via the web

  Built into the ISE's Project Navigator is the ability to search thousands of solution records found on the Xilinx software support home page from any error. Because the solutions are maintained on the Xilinx support home page, you are guaranteed to have the most up-to-date solutions available.

For a detailed description of key features, refer to the Key Features file by selecting **`Start`** → **`Programs`** → **`Xilinx Foundation Series ISE`** → **`Key Features`**.

Architecture Support

The software supports the following architecture families in this release.

- Spartan™ / XL / -II
- Virtex™ / -E / -II
- XC9500™ / XL / XV
- XC4000™E / L / EX / XL / XLA

Documentation

Xilinx provides a suite of documentation that includes detailed online help and online software manuals. The online software manuals are provided in both an HTML browser and PDF file formats.

The following online help is available:

- Project Navigator
- VHDL, Verilog and ABEL keywords
- Design entry tools (HDL Editor, Schematic Editor, Symbol Editor, and State Machine Editor)
- Design implementation tools (PROM File Formatter, FPGA Editor, Floorplanner, Hardware Debugger, JTAG Programmer, Flow Engine, Timing Analyzer, Constraints Editor, and LogiBLOX)
- Umbrella help

Online software manuals include the following:

- *CORE Generator User Guide*
- *Constraints Editor Guide*
- *Development System Reference Guide*
- *Floorplanner Guide*
- *FPGA Editor Guide*
- *Foundation Series ISE 3.1i User Guide*
- *Foundation Series ISE 3.1i Quick Start Guide*

- *Hardware Debugger Guide*
- *Hardware User Guide*
- *JTAG Programmer Guide*
- *Libraries Guide*
- *LogiBLOX Guide*
- *PROM File formatter Guide*
- *Synthesis and Simulation Design Guide*
- *Timing Analyzer Guide*
- *Synopsys VHDL Reference Guide (PDF only)*
- *Synopsys Verilog Reference Guide (PDF only)*
- *XST User Guide*

All of these manuals are supplied on the Documentation CD as well as the Design Environment CD. You can also access these manuals from the Xilinx Web site. The URL is `http://toolbox.xilinx.com/docsan`

You can also access the online software manuals from the Xilinx Web Site home page (`http://www.xilinx.com`) as follows:

1. Click **Service & Support**.
2. Click **Software Manuals**.

In addition to the online version of the software manuals, Xilinx also provides PDF versions for printing.

Two hard copy books are also shipped with the software: the *Foundation Series ISE 3.1i Quick Start Guide* and the *Foundation Series ISE Installation Guide and the Release Notes*.

Other manuals include the *MTI Reference Guide* (which resides on the CD that contains the MTI simulation software) as well as the StateCAD, StateBench, and the HDL Bencher User's Guides.

Chapter 2

Setting Up the Tools

This chapter includes a list of product configurations, general instructions for installing the software, customer support contacts, and licensing information.

For a detailed discussion of installation and licensing, refer to the *Foundation Series ISE Installation Guide and Release Notes*.

This chapter contains the following sections:

- "Product Configurations"
- "Installing Software"
- "Customer Service"
- "Technical Support"

Product Configurations

Following is a list of the product configurations for this release.

- Base-Express (DS-FSE-BSX-PC)
- Express (DS-FSE-EXP-PC)
- Elite (DS-FSE-ELI-PC)

All of these configurations contain FPGA Express, StateCAD Xilinx Edition, HDL Bencher Xilinx Edition, and the Modelsim Xilinx Edition (MXE) simulator. FPGA Express and Modelsim require licensing. Licensing for MXE is relatively straight-forward—follow the installation instructions described in the "Installing MXE Software" section.

To license FPGA Express, refer to the *Foundation Series ISE Installation Guide and Release Notes*.

Installing Software

Ensure the optimum use and operation of your new design tools by installing the software on the recommended hardware with sufficient memory (RAM and hard disk "swap" space). If you experience problems with either the installation, operation, or verification of your installation, contact the Xilinx Technical Support hotline. Refer to the "Technical Support" section of this chapter for specifics.

Please refer to the *Foundation ISE Installation Guide and Release Notes* for complete details on installation and prerequisites for installation.

Installing Xilinx Software

This subsection explains how to install the Xilinx software tools from the Xilinx Foundation Series ISE 3.3i CD. Note that this CD also contains the FPGA Express software.

1. Select **Start** → **Run**. Type **d:setup.exe** in the Open field of the Run window and click **OK**. (If your CD-ROM drive is not the "d" drive, substitute the appropriate drive designation.)
2. Follow the instructions on the screen to install the software. You will be asked to register the product from the Welcome screen during install. You can register via the web, email, or fax.

   In order to register the product, you need to provide the following information:

   - Product ID

     Your product ID number is located on the back of your software CD pack.
   - Your name
   - Company
   - Mailing address
   - Phone number
   - email address

   When you register, Xilinx gives you a Registration ID. You must have the registration ID in order to complete the installation.

The installer first installs all of the Xilinx software and then invokes the installer for FPGA Express. Make sure that you install FPGA Express in the default directory indicated. Your FPGA Express synthesis FlexLM license file will be emailed to you. When install is complete, remove the CD.

You may need to reboot your PC to allow the environment variables and path statement to take effect before you can run the design implementation tools. The Install program will inform you if you need to reboot.

Installing Documentation

The Documentation CD allows you to either install the documentation to your local disk or to view them directly from the CD. The Design Environment CD allows you to install the documentation to your local disk as part of the main software install procedure. Both CDs also contains PDF versions of manuals that can be viewed with the Adobe Acrobat reader.

Installing the documentation is optional, that is, it is not required to run the software.

To install the Xilinx documentation CD, insert the CD and follow the instructions. Alternatively, the documentation may be installed from the Design Environment CD when installing the Xilinx Foundation Series ISE 3.3i software.

Installing Test Bench, Simulation, and State Machine Software

HDL Bencher testbench generation software, StateCAD state diagram editor, and ModelSim Xilinx Edition simulation tools are available through separate installation programs outside of the main Foundation Series ISE installer. In order to complete the Basic Tutorial in Chapter 4, you must install the HDL Bencher and ModelSim software.

Note If you already have installed MTI and VSS versions of the HDL Bencher, ModelSim, and StateCAD, refer to the “Using Other Versions of ModelSim, HDL Bencher, and StateCAD” section.

Installing MXE Software

The ModelSim Xilinx Edition CD contains the ModelSim Xilinx Edition simulator from MTI. You must install a licensed version of Modelsim to complete the basic tutorial in Chapter 4. To install the CD, perform the following steps:

1. Insert the ModelSim Xilinx Edition CD.
2. Select **Start** → **Programs** → **Foundation Series ISE** → **Partner Products** →**Install Modelsim Xilinx Edition**.

   When you install this software, you are prompted for licensing. Follow the instructions on the screen to license and install the product.
3. Remove the CD when installation is complete.

Installing HDL Bencher and StateCAD Software

The Xilinx Foundation Series ISE 3.3i software includes HDL Bencher testbench generation software and StateCAD state diagram editor software.

1. Insert the Design Environment CD.
2. To install the HDL Bencher, select **Start** → **Programs** → **Foundation Series ISE** → **Partner Products** → **Install HDL Bencher Xilinx Edition**

   Follow the instructions on the screen to install the product.
3. The StateCAD software is not required to perform the Tutorial in Chapter 4. It is included with Foundation ISE to facilitate design entry, debug, and analysis of state machines. StateCAD includes a translation tool for importing designs originally created with the Xilinx Foundation State Editor. StateCAD significantly enhances state machine design through its FSM, and Logic Wizards. StateCAD generates FPGA optimized and error-free HDL. It also detects hundreds of logic errors and warnings up-front, and provides solutions for them before it generates any HDL.

   To install the StateCAD software, select **Start** → **Programs** → **Foundation Series ISE** → **Partner Products** → **Install StateCAD Xilinx Edition**.

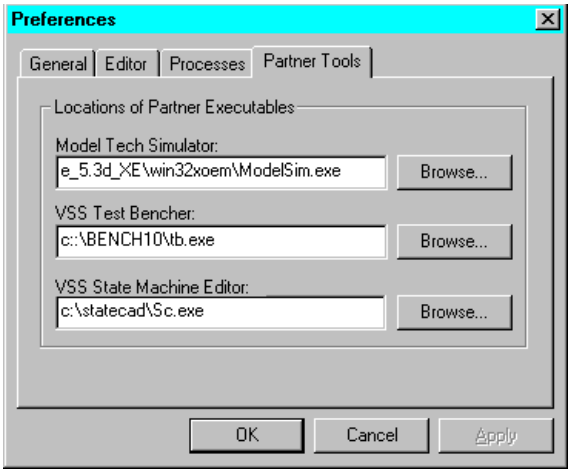
Follow the instructions on the screen to install the product.

Using Other Versions of ModelSim, HDL Bencher, and StateCAD

You can integrate non-Xilinx Editions of ModelSim, HDL Bencher, and StateCAD with ISE as follows.

1. Within the Project Navigator, select **`Edit`** → **`Preferences`** and then select the Partners tab.

   The following dialog box displays.



2. Enter the full path name to each of your tools or browse to the location of the desired executable (that is, ModelSim.exe for the Modeltec simulator, tb.exe for the VSS Test Bencher, and Sc.exe for the VSS State Machine Editor).

Customer Service

For software licensing information, warranty status, shipping, and order management issues, contact Xilinx Customer Service using the information in the following table.

Country	Telephone	Facsimile
United States and Canada[1]	1-800-624-4782	408-559-0115
United Kingdom[2]	01932-333550	01932-828521

Country	Telephone	Facsimile
Belgium[2]	0800 73738	
France[2]	0800 918333	
Germany[2]	0130 816027	
Italy[2]	1677 90403	
Netherlands[2]	0800 0221079	
Other European Locations[2]	(44) 1932-333550	(44) 1932-828521
Japan	81 3 3297 9153	81 3 3297 9189

[1] Mon-Fri, 8:00 am - 5:00 pm Pacific time

[2] Monday-Friday, 9:00 a.m. to 5:30 p.m. United Kingdom time—English speaking only.

If you are an international customer, contact your local sales representative for customer service issues. Refer to the Xilinx web site at http://support.xilinx.com/company/sales/int_reps.htm for contact information.

A complete list of Xilinx worldwide sales offices is at http://support.xilinx.com/company/sales/offices.htm.

Technical Support

The following section details how to reach the Xilinx Application Service centers for your area. If you experience problems with the installation or operation of your software, Xilinx suggests that you first go to our http://support.xilinx.com website.

You can also contact the Xilinx Technical Support hotline by phone, email, or fax. When e-mailing or faxing inquiries, provide your complete name, company name, and phone number. The following table gives Worldwide contact information for Xilinx Application Service centers.

Location	Telephone	Electronic Mail	Facsimile (Fax)
North America	1-408-879-5199 1-800-255-7778	hotline@xilinx.com	1-408-879-4442
United Kingdom	44-1932-820821	ukhelp@xilinx.com	44-1932-828522
France	33-1-3463-0100	frhelp@xilinx.com	33-1-3463-0959
Germany	49-89-93088-130	dlhelp@xilinx.com	49-89-93088-188
Japan	local distributor	jhotline@xilinx.com	local distributor
Korea	local distributor	korea@xilinx.com	local distributor
Hong Kong	local distributor	hongkong@xilinx.com	local distributor
Taiwan	local distributor	taiwan@xilinx.com	local distributor
Corporate Switchboard	1-408-559-7778		

Chapter 3

Software Overview

This overview describes the basic concepts and tools of the Foundation Series ISE release. For details on how to use the tools, refer to the "Basic Tutorial" chapter in this manual or the in-depth tutorial on the Web (`http://www.support.xilinx.com/support/techsup/tutorials/`).

This chapter contains the following sections.

- "Starting the Foundation Series ISE Software"
- "Project Navigator"
- "Design Entry"
- "Design Synthesis"
- "Design Constraints"
- "Functional Simulation"
- "Design Implementation"
- "Timing Simulation"
- "Device Programming"

Starting the Foundation Series ISE Software

To start the software, select **`Start`** → **`Programs`** → **`Foundation Series ISE`** → **`Project Navigator`**.

Project Navigator

The Project Navigator—the overall project management tool—contains the software tools used in the design process. Within the Project Navigator, you can access the following tools: design entry, synthesis, design implementation, device programming tools, and design verification which includes functional and timing simulation as well as static timing analysis.

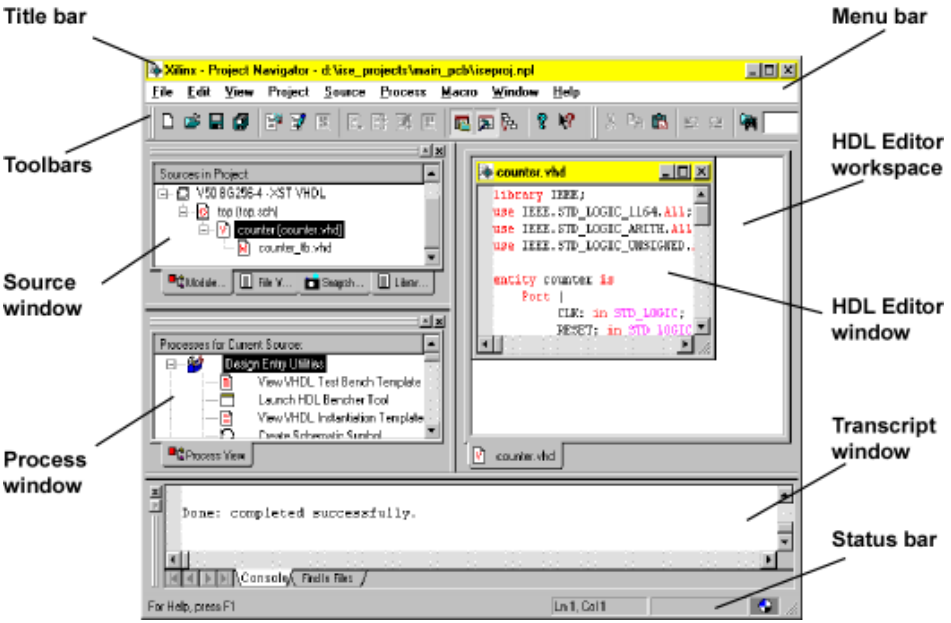


Figure 3-1 Project Navigator

The toolbar buttons at the top of the main window provide shortcuts for commonly used commands.

- The Source window contains the source modules for the project.
- The Process window contains the processes that are associated with the selected project source.
- The area located on the right side of the Project Manager is the HDL Editor window.

- The Transcript window at the bottom of the Project Navigator displays informational, warning, and error messages.
- The Status bar displays the status of the current process that is running.

For detailed information about the Project Navigator, refer to the "Project Navigator" chapter in the *Foundation Series ISE 3.1i User Guide*.

Project Navigator Online Help

The Project Navigator contains online help for each of the menu items located in the pulldown menus. You can access the help for these menu items by pressing F1.

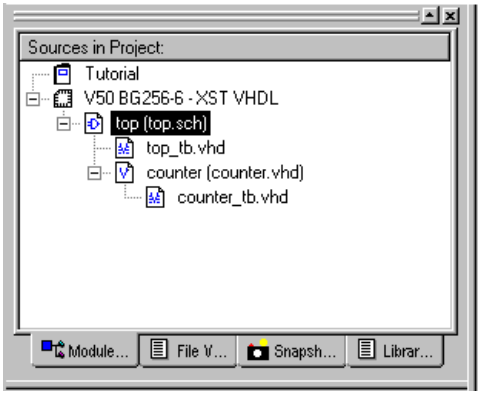
For example to access help for the New Source menu item from the Project menu, click the Project menu and then place the cursor on New Source. Press F1 to display the help.

Online help is also available for Project Navigator processes. To access, select a process and press F1.

Source Window

The Source window displays all of the design files associated with a project. A source is any element that contains information about a design. Sources include any files that are needed to describe the behavior of your design: schematics or HDL source files, test files, and design documentation.

The following figure shows an example of a Source window.



Icons represent the various types of source files. To view a complete list of source icons, click the context-sensitive Help icon and then click an unused area of the source window.

Double clicking a source invokes the editor for that source. For example, double clicking a schematic source file invokes the Schematic Editor; the schematic displays within the Schematic Editor tool. Correspondingly, double clicking an HDL file invokes the HDL Editor. When a source file's place in the design hierarchy is modified, the Source window automatically updates to reflect the change. Use the Project pulldown menu or toolbar button to perform the following operations on a source file:

- Create a New Source
- Add a new source to a project
- Copy a source to the current project folder
- Delete Implementation Data
- Archive a project
- Take a snapshot
- Apply Project Properties

Process Window

The Process window displays all the processes that apply for a selected source in the Source window. Processes include synthesis, test bench generation, simulation, implementation, device programming, report generation, or any other logical design step.

The Project Navigator's context-sensitivity automatically determines the design processes for any selected source. The processes are context-sensitive in two ways:

- The processes that display in the Process window depend on the source file that is highlighted in the Source window.
- Some of the processes that display in the Process window differ depending on the device type (FPGA or CPLD).

You run a design process by double clicking a process in the process window. By default the Project Navigator automatically updates outdated processes or reports.

A green check mark next to a process indicates that the process is up-to-date and completed without problems. A yellow exclamation point indicates that the process is up-to-date and completed with warnings. A red X indicates that a process failed due to errors.

You can also set properties for many of the processes. To set properties for a process, right click on the process and then select Properties from the menu or select the process and then select **`Process`** → **`Properties`**. For a description of all of the Implement Design properties, refer to the "Implementing the Design" chapter in the *Foundation Series ISE 3.1i User Guide*

For a description of the Synthesize properties for XST, see the *XST User Guide*.

Transcript Window

The Transcript window displays all messages, errors, and warnings that result from running processes. You can view this window or hide it by clicking the toolbar button.

When you run a process, messages display in the Transcript window indicating the status of the process. If your design contains synthesis errors or warnings, you can double-click the error or warning message that displays in the Transcript window and the source containing the error is opened. In the source, a red dot displays next to the line containing the error. Alternatively you can right click on the error and navigate to a solution record at support.xilinx.com.

Figure 3-2 Transcript Window

In the preceding figure, the Implement Design process has been run and the Transcript window indicates that the last process that ran was PAR (Place and Route --par.bat).

Design Entry

This section describes the design entry tools.

HDL Editor

The ISE language sensitive text editor for HDL includes color coding and context-sensitive help for reserved words. When color coding is enabled, different colored text is used for strings, comments, keywords, and directives. ISE contains language templates so that you can insert existing text structures into your source files.

The "Basic Tutorial" chapter explains how to use the language templates.

The HDL Editor supports Verilog, VHDL, and ABEL. You can customize the Editor's display, input and formatting options.

For details, see the "HDL Sources" chapter in the *Foundation Series ISE 3.1i User Guide*.

ECS Schematic Editor

The ECS (Engineering Capture System) is the Schematic Editor.

The ISE Schematic Editor includes the following features:

- Built-in design rule checking
- VHDL and Verilog structural netlist generation
- Matching symbol generation

For detailed information, refer to the "Schematic Sources" chapter in the *Foundation Series ISE 3.1i User Guide*.

Symbol Editor

You can use the Symbol Editor to create and edit symbols for the Schematic Editor.

Types of symbols that can be created include block symbols and graphic symbols. Block symbols are used to build hierarchical designs.

For detailed information, refer to the "Schematic Sources" chapter in the *Foundation Series ISE 3.1i User Guide*.

StateCAD and StateBench

The Foundation ISE software includes the Xilinx Edition of StateCAD ® and StateBench® for state machine designs. StateCAD includes a State Machine Wizard to help you develop the initial state machine, a Logic Wizard to create data flow structures, and an Optimization Wizard to maximize performance for the target device.

When installed, StateBench can automatically create VHDL test benches and Verilog test fixtures from StateCAD designs.

Refer to the StateCAD and StateBench online help and documentation for specific information on using these products. For installation information, refer to the "Setting Up the Tools" chapter in this manual or the *Foundation Series ISE Installation Guide and Release Notes*.

For information on how to launch the tool, refer to the "State Diagrams" chapter in the *Foundation Series ISE 3.1i User Guide*.

LogiBLOX and CORE Generator Modules

LogiBLOX is a design tool for creating high-level modules such as counters, shift registers, and multiplexers for FPGA and CPLD designs. LogiBLOX includes both a library of generic modules and a set of tools for customizing these modules. LogiBLOX modules are pre-optimized to take advantage of Xilinx architectural features such as Fast Carry Logic for arithmetic functions and on-chip RAM for dual-port and synchronous RAM. With LogiBLOX, you can create high-level LogiBLOX modules that will fit into your schematic-based design or HDL-based design. For detailed information, refer to the "LogiBLOX" chapter in the *Foundation Series ISE 3.1i User Guide*.

Xilinx CORE Generator is a design tool that creates parameterizable cores, optimized for Xilinx FPGAs. The CORE Generator library includes cores as complex as DSP filters and multipliers and as simple as delay elements. You can use these cores as building blocks in order to complete your design more quickly.

For details on how to instantiate cores, refer to the "CORE Generator" chapter in the *Foundation Series ISE 3.1i User Guide*. For complete information about CORE Generator, refer to the online manual, *CORE Generator User Guide*.

Design Synthesis

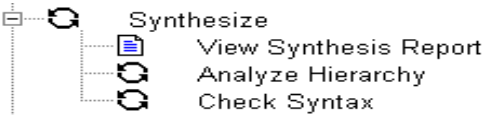
The Foundation ISE software contains two synthesis tools—XST and FPGA Express. Both synthesis engines accept the same types of input files and then generate the necessary output files so that you can place and route your design. The main difference is that XST can only synthesize designs that are either all VHDL or all Verilog. FPGA Express allows a mixture of both VHDL and Verilog. The following subsections describe each tool.

XST

You can use XST if designing with the following device families.

- Virtex
- Virtex2
- VirtexE
- Spartan2
- SpartanXL
- XC9500
- XC9500XL
- XC9500XV

All synthesis processes are placed under a process in the Process window called Synthesize. This process runs the entire synthesis flow to generate the required files so that you can place and route your design. The following diagram illustrates the synthesis flow in the Process window for XST.



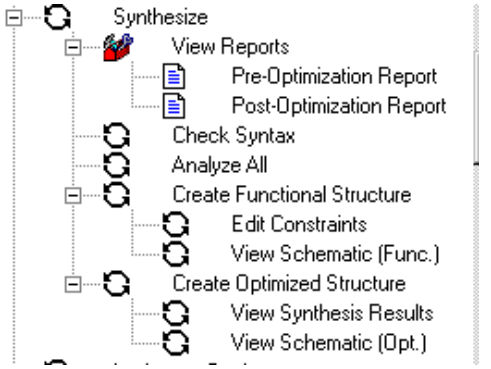
You can also set various properties for XST synthesis by right clicking Synthesize and then selecting Properties from the menu. The Process Properties dialog box displays allowing you to select Synthesis Options, HDL Options, and Xilinx Specific Options.

For detailed information about XST, refer to the *XST User Guide*.

FPGA Express

FPGA Express can synthesize all devices supported for this release.

The Synthesize process runs the complete synthesis flow to produce the necessary files to place and route your design. The following diagram illustrates the synthesis flow in the process window for FPGA Express.



You can examine static timing results with the Express Time Tracker after synthesis and before implementation as follows:

Note You must be licensed (DS-FSE-EXP-PC or DS-FSE-ELI-PC) to use FPGA Express to access the Time Tracker and the Express Constraints Editor.

1. Double-click View Synthesis Results in the Process window.
2. Select the Paths tab from the Time Tracker to view estimated delays.

Design Constraints

With the design implementation tools, you can control the implementation of a design by entering constraints. There are two basic types of constraints that you can apply to a design: location constraints and timing constraints.

Location constraints control the mapping and positioning of the logic elements in the target device. The most common location constraints are pad constraints. They are used to lock the pins of the design to specific I/O locations so that the pin placement is consistent from revision to revision.

Timing constraints tell the software which paths are critical, and therefore, need closer placement and faster routing. Conversely, timing constraints also tell the software which paths are not critical and, therefore, do not need closer placement or faster routing. Both the placer and the router can be timing constraint driven.

You can enter constraints at various phases of the design process:

- Schematic Editor
- HDL Editor
- UCF file

  You can enter constraints manually or with the Xilinx Constraints Editor. Refer to the *Constraints Editor Guide*.
- FPGA Express
- XST constraints file
- Floorplanner

  Refer to the *Floorplanner Guide* for details.

Functional Simulation

Functional Simulation allows you to verify the logic of your design before you synthesize it. You can observe the circuit's behavior at its inputs and outputs as well as the behavior of internal nodes. You can use test benches (VHDL) or test fixtures (Verilog) to specify circuit input stimuli and expected output responses and then test to those specifications prior to synthesis.

Foundation ISE supports RTL (Register Transfer Level) simulation as its functional simulation. RTL simulation is performed after design entry but prior to synthesis.

The "Basic Tutorial" chapter has an example of a functional simulation for a counter.

For more detailed information about functional simulation, refer to the "Simulation" chapter in the *Foundation Series ISE 3.1i User Guide*.

Design Implementation

After synthesis is complete, the implementation tools perform the translate map, place, route, (fit for CPLDs), and programming file generation phases of the design flow.

The Process window first step, Translate, merges all of the input netlists. This is accomplished by running NGDBuild. For a complete description of NGDBuild, refer to the "NGDBuild" chapter in the *Development System Reference Guide*.

For FPGAs, the next step is the technology mapper. Map optimizes the gates and trims unused logic in the merged NGD netlist. This step also maps the design's logic resources; logic in the design is mapped to resources on the silicon, and a physical design rule check is performed. For more information about MAP, refer to the "MAP—The Technology Mapper" chapter in the *Development System Reference Guide*.

After the FPGA design is mapped, it is placed and routed. In the place stage, all logic blocks, including the configurable logic blocks (CLB) and input/output blocks (IOB) structures, are assigned to specific locations on the die.

If timing constraints have been placed on particular logic components, the placer tries to meet those constraints by moving the corresponding logic blocks closer together.

In the routing stage, the logic blocks are assigned specific interconnect elements on the die. If timing constraints have been placed on particular logic components, the router tries to meet those constraints by choosing a faster interconnect. For more information about PAR, refer to the "PAR—Place and Route" chapter in the online software document, *Development System Reference Guide*.

The CPLD fitter implements designs for the XC9500, XC9500XV, and XC9500XL devices. The fitter outputs several files: fitting report (*design_name*.rpt), static timing report (*design_name*.tim), guide file (*design_name*.gyd, programming file (*design_name*.jed), and timing simulation database (*design_name*.nga).

For detailed information about implementing CPLD designs, refer to the Foundation online help.

For more detailed information about design implementation, refer to the "Implementing the Design" chapter in the *Foundation Series ISE 3.1i User Guide*.

Interpreting the Reports

The reports generated by implementation provide information on logic trimming, logic optimization, timing constraint performance, and I/O pin assignment. To access a report, double-click the report in the Process window after you have run your design through design implementation.

Translation Report

The translation report (.bld) contains warning and error messages from the three translation processes: conversion of the EDIF netlist to the Xilinx NGD netlist format, timing specification checks, and logical design rule checks. The report lists the following:

- Missing or untranslatable hierarchical blocks
- Invalid or incomplete timing constraints
- Output contention, loadless outputs, and sourceless inputs

Map Report (FPGAs)

The Map Report (.mrp) contains warning and error messages detailing logic optimization and problems in mapping logic to physical resources. The report lists the following information:

- Removed logic. Sourceless and loadless signals can cause a whole chain of logic to be removed. Each deleted element is listed with progressive indentation, so the origins of removed logic sections are easily identifiable; their deletion statements are not indented.
- Logic that has been added or expanded to optimize speed.

- The Design Summary section lists the number and percentage of used CLBs, IOBs, flip-flops, and latches. It also lists occurrences of architecturally-specific resources like global buffers and boundary scan logic.

Note The Map Report can be very large. To find information, use key word searches. To quickly locate major sections, search for the string '---', because each section heading is underlined with dashes.

Pre-Route Static Timing Report (FPGAs Only)

Pre-route static timing reports can be very useful in evaluating timing performance. Although route delays are not accounted for, the logic delays can provide valuable information about the design.

If logic delays account for a significant portion (> 50%) of the total allowable delay of a path, the path may not be able to meet your timing requirements when routing delays are added.

Routing delays typically account for 40% to 60% of the total path delays. By identifying problem paths, you can mitigate potential problems before investing time in place and route. You can redesign the logic paths to use less levels of logic, tag the paths for specialized routing resources, move to a faster device, or allocate more time for the path.

If logic-only-delays account for much less (<35%) than the total allowable delay for a path or timing constraint, then the place-and-route software can use very low placement effort levels. In these cases, reducing effort levels allow you to decrease runtimes while still meeting performance requirements.

Place and Route Report (FPGAs)

The Place and Route Report (.par) contains the following information.

- The overall placer score which measures the "goodness" of the placement. Lower is better. The score is strongly dependent on the nature of the design and the physical part that is being targeted, so meaningful score comparisons can only be made between iterations of the same design targeted for the same part.

- The Number of Signals Not Completely Routed should be zero for a completely implemented design. If non-zero, you may be able to improve results by using re-entrant routing or the multi-pass place and route flow.
- The timing summary at the end of the report details the design's delays.

Pad Report (FPGAs)

The Pad Report lists the design's pinout in three ways.

- Signals are referenced according to pad numbers.
- Pad numbers are referenced according to signal names.
- PCF file constraints are listed.

Asynchronous Delay Report (FPGAs)

This report shows the 20 worst net delays within the design.

Fitting Report (CPLDs)

The Fitting Report (*design_name*.rpt) lists summary and detailed information about the logic and I/O pin resources used by the design, including the pinout, error and warning messages, and Boolean equations representing the implemented logic.

Post Route Timing Report (FPGAs Only)

Post-Route timing reports incorporate all delays to provide a comprehensive timing summary. If a placed and routed design has met all of your timing constraints, then you can proceed by creating programming data and downloading to a device. On the other hand, if you identify problems in the timing reports, you can try fixing the problems by increasing the placer effort level, using re-entrant routing, or using multi-pass place and route. You can also redesign the logic paths to use fewer levels of logic, tag the paths for specialized routing resources, move to a faster device, or allocate more time for the paths.

Timing Report (CPLDs)

The report is equivalent to the Post Route Timing Report—provides a comprehensive timing summary of all delays.

Lock Pins Report

The Lock Pins Report is generated by running PIN2UCF. PIN2UCF is a program that generates pin locking constraints in a UCF file by reading a placed NCD file for FPGAs or GYD file for CPLDs. PIN2UCF writes its output to an existing UCF file. If there is no existing UCF file, PIN2UCF creates a new file.

The Lock Pins Report file has two sections: Constraint Conflicts Information and List of Errors and Warnings.

- The Constraints Conflicts Information section does not display if there are fatal input errors, for example, missing inputs or invalid inputs. However, the created report file contains the List of Errors and Warnings.
- The Constraints Conflicts Information section has two subsections:
  - Net name conflicts on the pins
  - Pin name conflicts on the nets

    If there are no conflicting constraints, both subsections under the Constraint Conflicts Information section contain a single line indicating that there are no conflicts.

The List of Errors and Warnings displays only if there are errors or warnings.

For detailed information about PIN2UCF, refer to the “PIN2UCF” chapter in the *Development System Reference Guide.*

MPPR Report (FPGAs Only)

The MPPR (Multi-Pass Place and Route) report is generated for FPGAs only. MPPR runs iterations of PAR with different cost tables. MPPR scores each PAR iteration and uses the scores to determine the best passes to save. Scores are based on the number of unrouted nets, delays on nets, and timing constraints. For details about MPPR, refer to the “Output from PAR” section in the *Development System Reference Guide*.

Other Design Implementation Tools

This section briefly discusses other design implementation tools.

Floorplanner (FPGAs)

The Floorplanner is a graphical placement tool that gives you control over placing a design into a target FPGA using a “drag and drop” paradigm with the mouse pointer.

The Floorplanner displays a hierarchical representation of the design in the Design Hierarchy window using hierarchy structure lines and colors to distinguish the different hierarchical levels. The Floorplan window displays the floorplan of the target device into which you place logic from the hierarchy.

Floorplanning is an optional methodology to help you improve performance and density of a fully, automatically placed and routed design. Floorplanning is particularly useful on structured designs and data path logic. With the Floorplanner, you see where to place logic in the floorplan for optimal results, placing data paths exactly at the desired location on the die.

With the Floorplanner, you can floorplan your design prior to or after running PAR. In an iterative design flow, you floorplan and place and route, interactively. You can modify the logic placement in the Floorplan window as often as necessary to achieve your design goals. You can save the iterations of your floorplanned design to use later as a constraints file for MAP.

For detailed information about the Floorplanner, refer to the *Floorplanner Guide*.

FPGA Editor

The FPGA Editor is a graphical application for displaying and configuring Field Programmable Gate Arrays (FPGAs). You can use this application to place and route critical components before running the automatic place and route tools on your design. You can also use the FPGA Editor to manually finish placement and routing if the routing program does not completely route your design.

The FPGA Editor requires a Native Circuit Description (NCD) file. This file contains the logic of your design mapped to components (such as CLBs and IOBs). In addition, the FPGA Editor reads from and writes to a Physical Constraints File (PCF).

For detailed information about the FPGA Editor, refer to the *FPGA Editor Guide*.

Post Fit ChipViewer (CPLDs)

You can use the ChipViewer to display a graphical representation of how the logic circuitry and I/Os are assigned to the CPLD macrocells and pins.

Timing Simulation

For Foundation ISE, timing simulation is a *gate-level* simulation that includes detailed timing information for the targeted device. Gate-level simulation is performed after synthesis and place and route.

You can run timing simulation by creating a test bench file with HDL Bencher graphically (no scripting required) or by modifying a template file (View VHDL or Verilog Test Bench Template).

Device Programming

This section explains how the bitstream file is created for FPGAs and CPLDs, and also describes which tools to use for downloading a bitstream to a device.

Create Programming File (FPGAs)

Double clicking Create Programming File runs BitGen. BitGen translates the physical implementation into a programming file (bit) that is used to program the FPGA. The BitGen executable creates the programming file. To set options, right click Create Programming Files and then select Properties to display the Process Properties dialog box.

For more information about the BitGen executable, refer to the "BitGen" chapter in the online software manual, *Development System Reference Guide*.

Viewing Programming File Generation Report (FPGAs)

This report contains information about the BitGen run.

PROM File Formatter (FPGAs)

An FPGA or daisy chain of FPGAs can be configured from serial or parallel PROMs. The PROM File Formatter can create MCS, EXO, or TEK style files. The files are read by a PROM programmer that turns the image into a PROM.

A HEX file can also be used to configure an FPGA or a daisy chain of FPGAs through a microprocessor. The file is stored as a data structure in the microprocessor boot-up code.

Hardware Debugger (FPGAs)

The Hardware Debugger can download a BIT file or a PROM file: MCS, EXO, or TEK file formats. A BIT file contains configuration information for an FPGA device. For more information on using the Hardware Debugger, see the *Hardware Debugger Guide*.

JTAG Programmer

For CPLDs, you generate the bitstream file by first highlighting the Part Number Flow Type line in the Sources window and then double-click on the JTAG Programmer process. You can then download the bitstream from your PC using the JTAG Programmer.

The JTAG Programmer software can be used to configure both FPGAs and CPLDs and supports both the XChecker and the Parallel Cable III. This is a GUI based program. See the *JTAG Programmer Guide* for details. Also, see the *Hardware User Guide* for information about cable compatibility.

Chapter 4

Basic Tutorial

This basic tutorial describes how to use the VHDL and schematic design entry tools, explains how to perform functional and timing simulation, and describes how to implement a design. For an in-depth explanation of the ISE design tools, see the in-depth tutorial on the Xilinx web site (`http://www.support.xilinx.com/support/techsup/tutorials/`).

The chapter contains the following sections.

- "Introduction"
- "Online Help"
- "Hints"
- "Design Entry (VHDL)"
- "Functional Simulation"
- "Design Entry (Top-Level Schematic)"
- "Design Implementation"
- "Timing Simulation"

Introduction

This tutorial accomplishes the following:

- Provides some helpful tips on how to use the Schematic Editor
- Describes how to create a VHDL module for a 4-bit counter
- Explains how to use the ISE simulation tools to perform a functional simulation of the 4-bit counter
- Describes how to create a schematic symbol for the VHDL module 4-bit counter

- Illustrates how to create a top-level schematic and instantiate VHDL modules into the schematic
- Explains how to use the Schematic Editor to wire the components together, add net names to the wires, create buses, and add I/O markers
- Describes how to implement the design and view the placed and routed design in the Floorplanner
- Explains how to perform a timing simulation of the top-level design.

Online Help

F1-On-Line Help is available and is context sensitive. For example, if you are using the Schematic Editor to add wires, press F1 to display help on adding wires.

Hints

The following subsections provide some useful tips about the Source and Process windows as well as the Schematic Editor.

Source and Process Windows

The Process window shows the processes that can be performed on the source selected in the Source window. The list of processes changes according to the type of source that is selected.

For example, to make a schematic symbol for a VHDL module, select the VHDL module in the Source window and then double-click the Create Schematic Symbol process in the Design Entries Utilities branch in the Process window.

Schematic Editor Tips

The following subsections provide some useful tips for using the Schematic Editor. To open the Schematic Editor, perform the following steps.

1. Select **`Project`** → **`New Source...`**.
2. Select Schematic from the list box. Enter a name in the File Name box.

3. Select **Next**.
4. Select **Finish**.

General Object-Action vs. Action-Object

The Schematic Editor is designed so that you can select the action you wish to perform followed by the object the action is to be performed on (action-object). In general, most Windows applications currently operate by selecting the object and then the action to be performed on that object (object-action). The following subsections illustrate how the action-object model works.

Dragging a Net Name

When dragging a net name, select **Edit** → **Drag** and then place the cursor on the wire associated with the net name in the center of the net name as shown in the figure below.

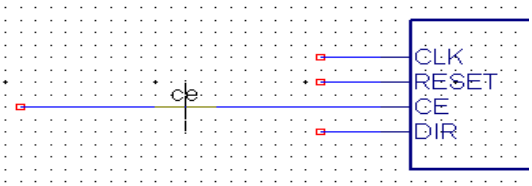


Figure 4-1 Dragging a Net Name

Adding a Net Name to an I/O Wire

To add net names to wires that are I/Os, perform the following steps.

1. Extend the length of each I/0 using **Add** → **Wire**.
2. Select **Add** → **Net Name** and position the cursor at the end of the wire when placing the net name as shown in the following figure.

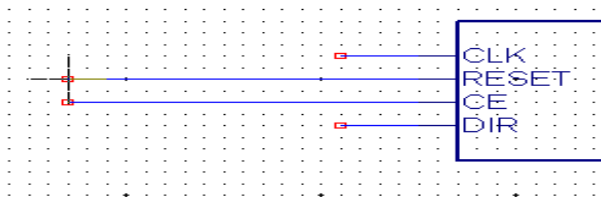


Figure 4-2 Adding a Net Name

3. Type the net name and press Enter.
4. Click the end of the wire to place the net name.

Adding I/O Markers

To add I/O markers, perform the following steps.

1. Select **Add** → **I/O Marker**.
2. Select the type of marker and either click the end of the wire or click and drag around a number of I/O wires to place the I/O markers.

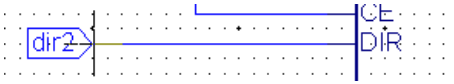


Figure 4-3 Adding I/O Markers

Design Entry (VHDL)

This section explains how to create a 4-bit counter module using the Language Templates.

Starting the ISE Software

To start ISE, select **Start** → **Programs** → **Foundation Series ISE 31i** → **Project Navigator** from the Start menu.

Creating a New Project

The following steps explain how to create a new project.

1. Select **File** → **New Project...**. The New Project dialog box displays.

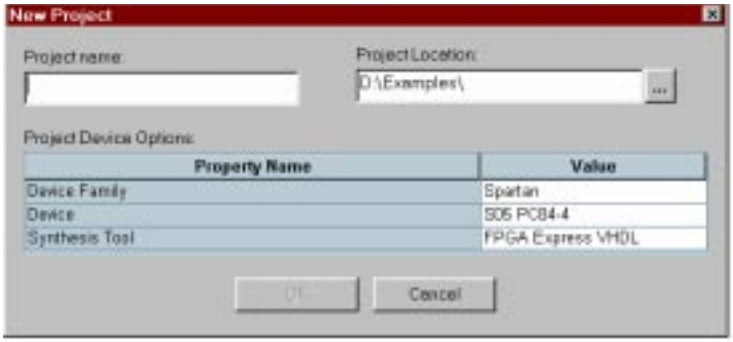


Figure 4-4 New Project Dialog Box

2. In the Project Location field, browse to the directory under which you want to create your new project directory. There is a browse button next to the Project Location field.
3. Create a new project by entering “Tutorial” in the Project Name field. When you enter “Tutorial” in the Project Name field, a Tutorial directory is automatically created in the directory path in the Project Location field. For example, for the directory path D:\My Projects, entering the Project Name Tutorial modifies the path to be D:\My Projects\Tutorial.
4. Use the pulldown arrow to enter the Value for each Property Name.

   There is a pulldown arrow in each Value field. However, you cannot see the arrow until you click in the field.

   Change the Values as follows:

   - Device Family: Virtex
   - Device: V50 BG256-6
   - Synthesis Tool: XST VHDL
5. Click **OK**. Foundation ISE creates a subdirectory and a new project.

Creating a 4-Bit Counter Module

Use the Language Templates to create a VHDL module for a counter as follows:

1. Select **Project** → **New Source**...
2. Select **VHDL Module** as the source type and give it a file name "counter".
3. Click **Next**.
4. Click **Next**.
5. Click **Finish** to complete the new source file template. An editor window displays showing the library declaration and use statements along with the entity and architecture pair for the counter.
6. Open the Language Templates by selecting **Edit** → **Language Templates** or by clicking the light bulb icon located on the far right on the toolbar.

7. In the Language Templates window, click the + sign next to VHDL and then click the + sign next to Synthesis Templates.
8. Click and drag the Counter template from the **VHDL** → **Synthesis Templates** folder to the counter architecture between the begin and end behavioral statements. Close the Language Template.

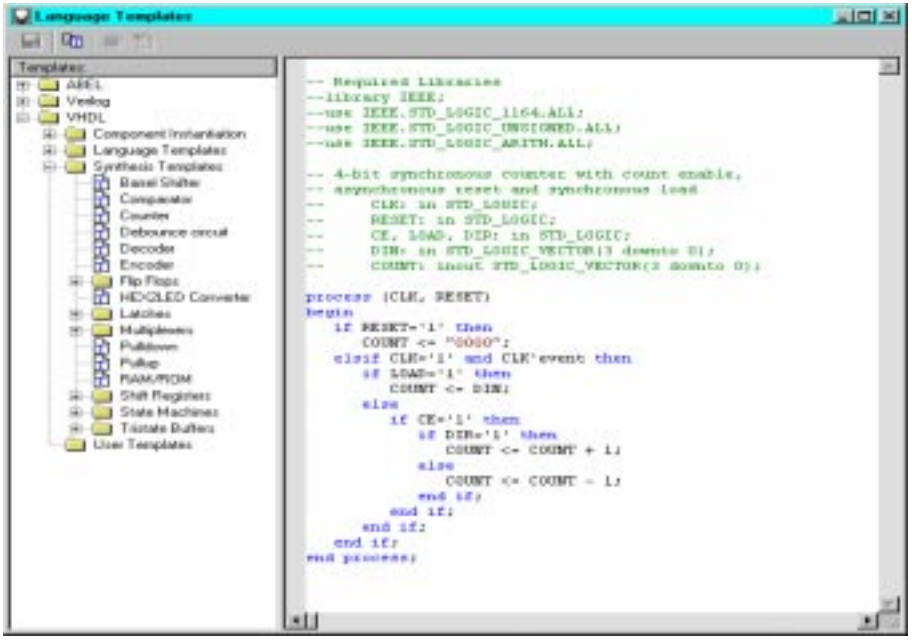


Figure 4-5 Language Templates

9. Cut and paste the port definitions from the comment section of the counter.vhd file into the parentheses in the port declaration of the counter entity. Uncomment the lines. A comment line has two dashes at the beginning of a line. To uncomment a line, remove the dashes. Make sure to remove the last semicolon after the COUNT port definition (that is, 3 downto 0). Following are the port definition lines to cut and paste:

```
-- CLK: in STD_LOGIC;
-- RESET: in STD_LOGIC;
-- CE, LOAD, DIR: in STD_LOGIC;
-- DIN: in STD_LOGIC_Vector(3 downto 0);
-- COUNT: inout STD_LOGIC_VECTOR(3 downto 0);
```

10. Delete the following commented lines.

```
-- Required Libraries
-- library IEEE;
-- use IEEE.STD_LOGIC_1164.ALL;
-- use IEEE.STD_LOGIC_ARITH.ALL;
-- use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
-- 4-bit synchronous counter with count enable,
-- asynchronous reset and synchronous load
```

11. Remove “Load,” from the following line underneath the Port declarations.

```
CE, LOAD, DIR: in STD_LOGIC;
```

12. Remove the following line beginning with DIN underneath the Port declarations.

```
DIN: in STD_LOGIC_VECTOR(3 downto 0);
```

13. Remove the following lines underneath the “begin” declaration:

```
if LOAD='1' then
  COUNT <= DIN;
else
```

14. Remove the second to the last “endif” statement. The following code shows which one to remove:

```
    endif
  endif
 endif (Remove this one)
endif
```

Your counter.vhd source should look like the following:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use  IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter is
    Port (
      CLK: in STD_LOGIC;
      RESET: in STD_LOGIC;
      CE,DIR: in STD_LOGIC;
      COUNT: inout STD_LOGIC_VECTOR(3 downto 0)
);
end counter;

architecture behavioral of counter is
```

```
begin

process (CLK, RESET)
begin
  if RESET='1' then
    COUNT <= "0000";
  elsif CLK='1' and CLK'event then
      if CE='1' then
        if DIR='1' then
          COUNT <= COUNT + 1;
        else
          COUNT <= COUNT - 1;
        end if;
      end if;
  end if;
end process;

end behavioral;
```

15. Save counter.vhd by selecting **File** → **Save** and then minimize the Project Navigator.

Functional Simulation

This section explains how to create a test bench using the HDL Bencher and then simulate the counter using ModelSim Xilinx Edition (MXE)

Creating a Test Bench with HDL Bencher

To create a test bench, perform the following steps.

1. Select counter (counter.vhd) in the Source window.
2. Double-click Launch HDL Bencher Tool in the Process window.
3. Click **OK** to use the default timing constraints for the test bench. Your screen should look like the following:

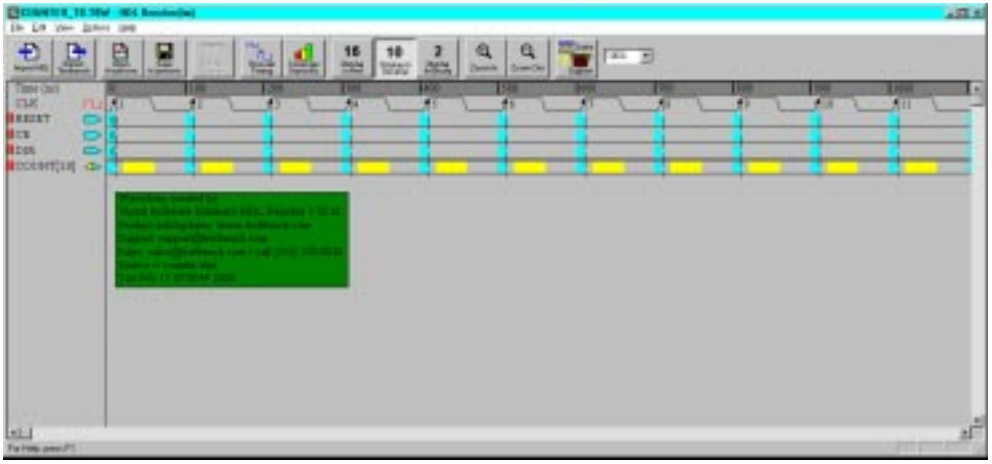


Figure 4-6 HDL Bencher Default Values

Note The blue areas are for entering input stimulus and the yellow areas are for entering expected response.

4. Initialize the counter as follows:

   a) Click the RESET cell just to the right of the blue cell in CLK cycle 1 until the cell is set to high.

   b) Click the RESET cell just to the right of the blue cell in CLK cycle 2 until the cell is reset to low.

   c) Click the CE cell just to the right of the blue cell in CLK cycle 3 until it is set to high and enable the counter.

   d) Click the DIR cell just to the right of the blue cell in CLK cycle 2 until the cell is set to high.

5. Enter the expected response as follows:

   a) Click the yellow COUNT[3:0] cell in CLK cycle 2 and click the Pattern button to launch the Pattern Wizard.

   b) Set the pattern wizard parameters so that the expected output counts from 0 to 7 as follows:

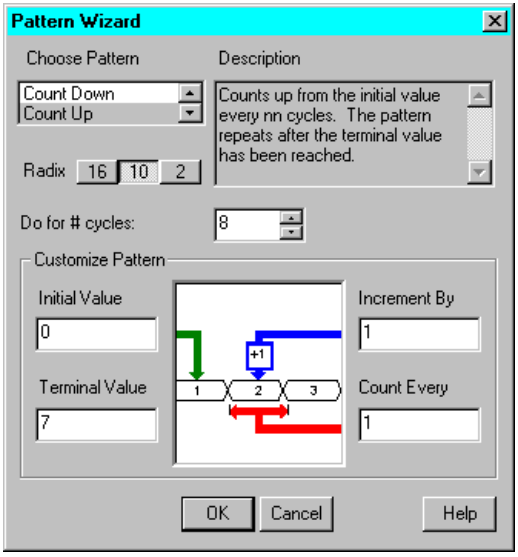


Figure 4-7 Pattern Wizard Settings

c) Click **OK** to complete the stimulus and response entry.

Your HDL Bencher window should look like the following:

Figure 4-8 HDL Bencher Stimulus and Response Entries

6. Click the Export Testbench icon to create the test bench.

7. Close the Edit Test Bench window.
8. Maximize Project Navigator.

Adding the Test Bench File to the Project

This subsection explains how to add the test bench file to the project.

1. Add the test bench to the project by selecting **Project** → **Add Source**.
2. Select the test bench file COUNTER_TB.VHD and click Open.
3. Select VHDL Test Bench and click **OK**.

   The Project Navigator source window should look like the following:

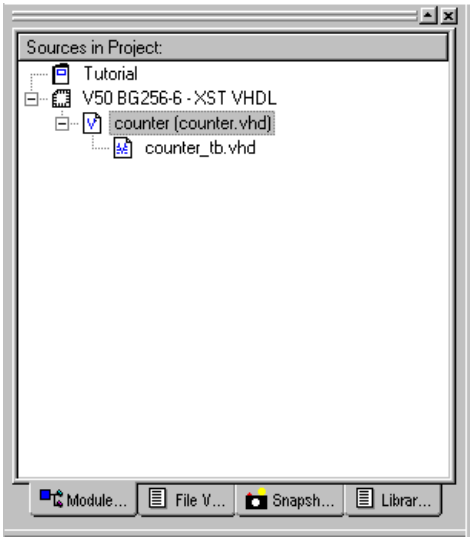


Figure 4-9 Test Bench File in Source Window

Simulating with ModelSim

To simulate with ModelSim, perform the following steps.

1. Select counter_tb.vhd in the Project Navigator Source window and double click Simulate Functional VHDL Model in the process window. This will bring up the ModelSim screen.

   Project Navigator creates a simulation macro file (do file) called counter_tb.fdo that does the following:

- Creates the design library
- Compiles the design and test bench source files
- Invokes the simulator
- Opens all the viewing windows
- Adds all the signals to the Wave window
- Adds all the signals to the List window
- Runs the simulation for the designated time

2. Click Run ModelSim if the simulation does not run automatically.
3. When the wave window displays, zoom out until the output waveform looks like the following:

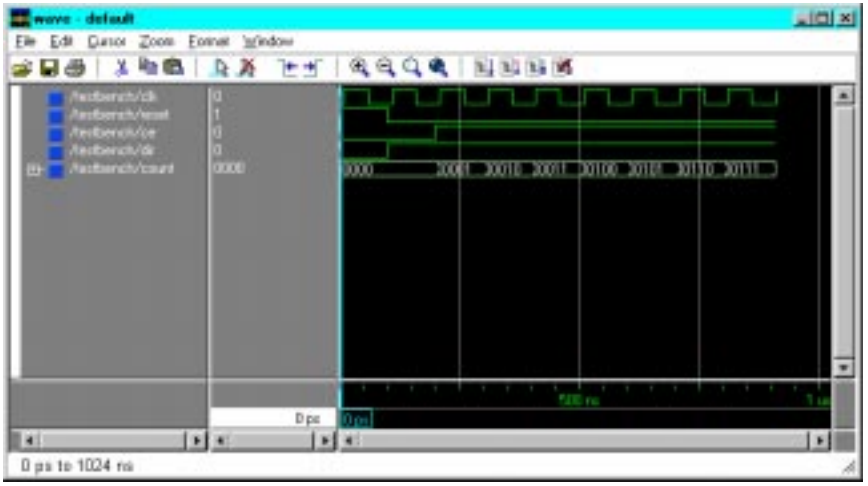


Figure 4-10 Functional Simulation Waveform

You have just created a counter in VHDL, created a test bench for the counter, and verified its functionality. Next you will instantiate two of these counter modules into a top-level schematic.

4. Close ModelSim.

Design Entry (Top-Level Schematic)

This section explains how to create a top-level schematic that contains instantiations of the counter module. Then this section describes how to wire together the modules, add net names to the wires, and add I/O markers.

Creating a Schematic Symbol for the VHDL Module

To create a schematic symbol for the VHDL module, perform the following steps:

1. Select your counter module (counter.vhd) in the Source window.

   Notice that the processes in the Process window change according to the type of source file selected in the Source window.

2. Double-click the Create Schematic Symbol process as shown in the following figure.

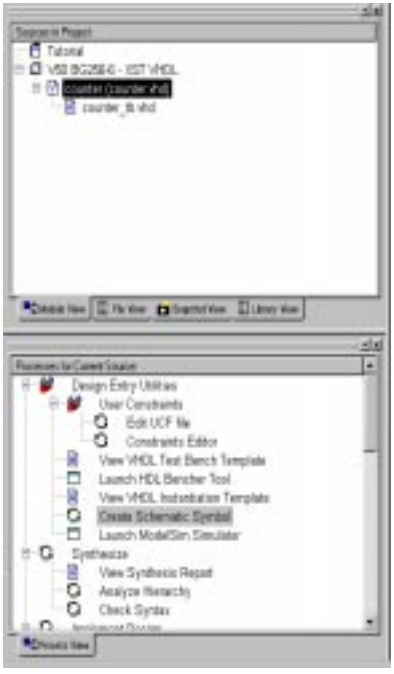


Figure 4-11 Create Symbol Process

Creating a New Top-Level Schematic

To create a new top-level schematic, perform the following steps.

1. Select **Project** → **New Source...** from the Project menu.
2. Select **Schematic** as the source type and name it “top”.
3. Click **Next** and then click **Finish**.

   A blank sheet opens in the Schematic Editor.

Instantiating VHDL Modules

Perform the following steps to instantiate VHDL modules into the top-level schematic.

1. Make the drawing toolbar visible by selecting it in the View menu.

Figure 4-12 Drawing Toolbar

2. Place two counter modules on the schematic by selecting **Add** → **Symbol...** from the menu or by clicking the Add Symbol button in the Drawing Toolbar.

3. Select counter from the Symbol Libraries window and place two counters in the schematic. Your schematic should look like the following diagram:

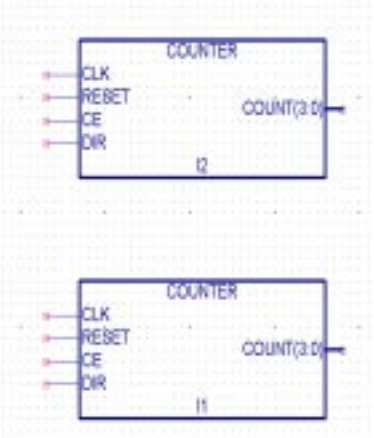


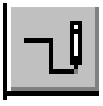
Figure 4-13 Instantiated VHDL Modules

4. Press Esc to exit the Add Symbols mode.

Wiring the Schematic

To interconnect the VHDL modules, perform the following steps.

1. Select the Add Wire tool from the Drawing Toolbar or select **Add** → **Wire** from the menu.



To add a wire between two schematic symbols, click once on the symbol pin, once at each vertex and once on the destination pin.

To add a hanging wire, click on the symbol pin to start the wire and then double-click at the location you want the wire to terminate. Make sure you add hanging wires for each pin on both counters.

2. Wire the symbols similar to the following schematic:

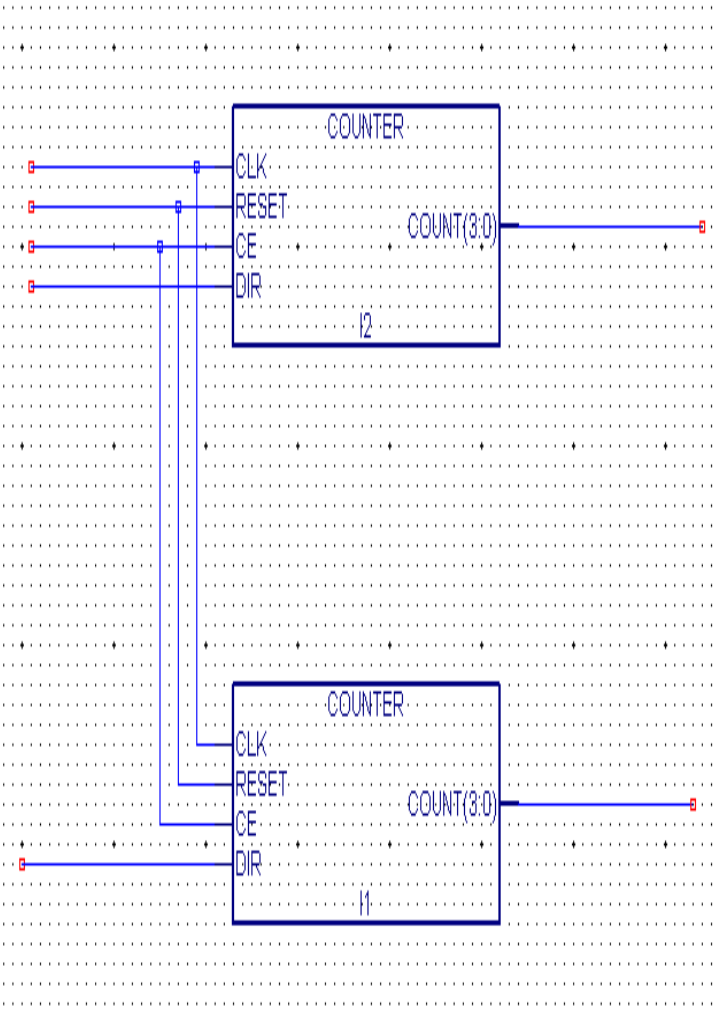


Figure 4-14 Interconnected Modules

Adding Net Names to Wires

To add net names to the wires, perform the following steps.

Note If you have not already extended the length of each I/O pin as explained in Steps 2 and 3, then extend the length of each of these wires before adding a net name.

1. Select the Add Net Names tool from the Drawing Toolbar.

2. From the keyboard, type the net name followed by the Enter key.
3. Place the net name on the end of the hanging wire as shown below and then click the left mouse button:

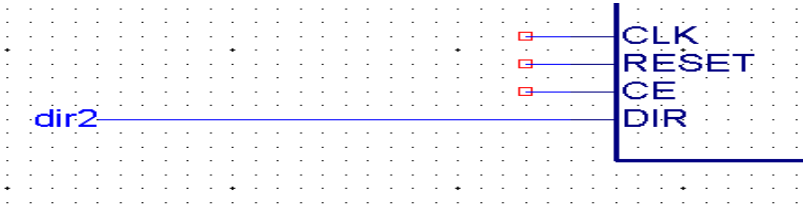


Figure 4-15 Net Name Example for I/Os

4. Finish adding net names so your schematic looks like the following diagram:

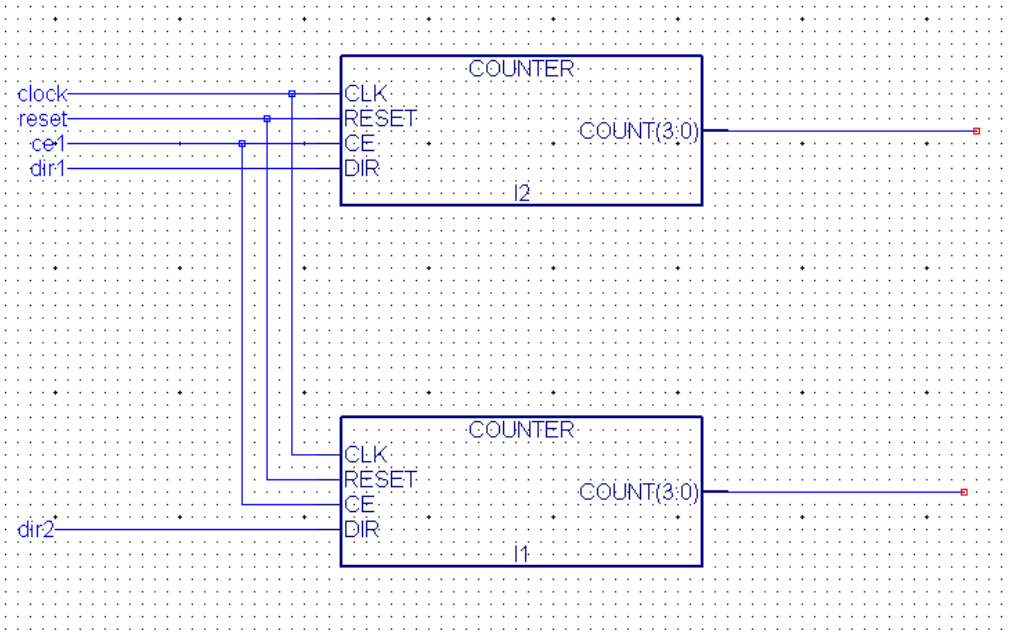


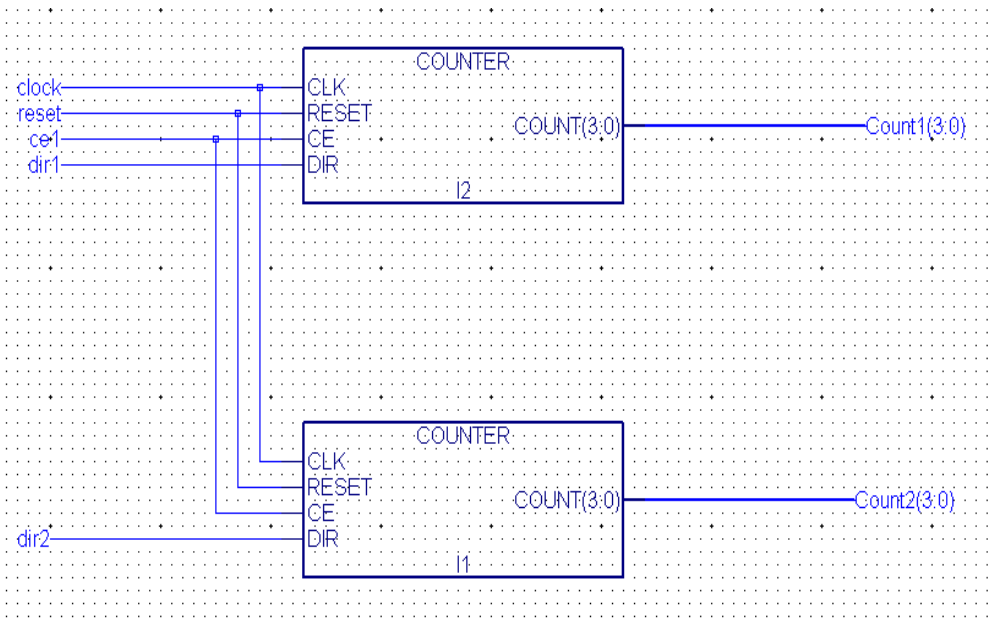
Figure 4-16 Net Names

Creating Buses

It is important to note that you can only create buses from existing individual wires. The following procedure explains how to create buses for Count1[3:0] and Count2[3:0].

1. Select the Add Net Names tool from the Drawing Toolbar.

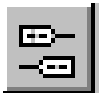
2. From the keyboard, type the bus name and size (for example, Count1[3:0]). Press Enter and then place the bus name on the end of the wire and then click the left mouse button.
3. Repeat Step 2 for the output of the second counter by naming it Count2[3:0].



Adding I/O Markers

To add I/O markers, perform the following steps.

1. Select the Add I/O Marker tool from the Drawing Toolbox.



2. With the Input type selected, click and drag around all the inputs that you want to add input markers to.

   Note To add individual markers click on the end of the hanging wire as shown below.

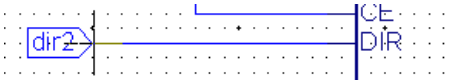


Figure 4-17 Adding Individual Markers

3. Change the I/O marker type to bidirectional and add markers to the counter outputs.

Your completed schematic should look like the following drawing:

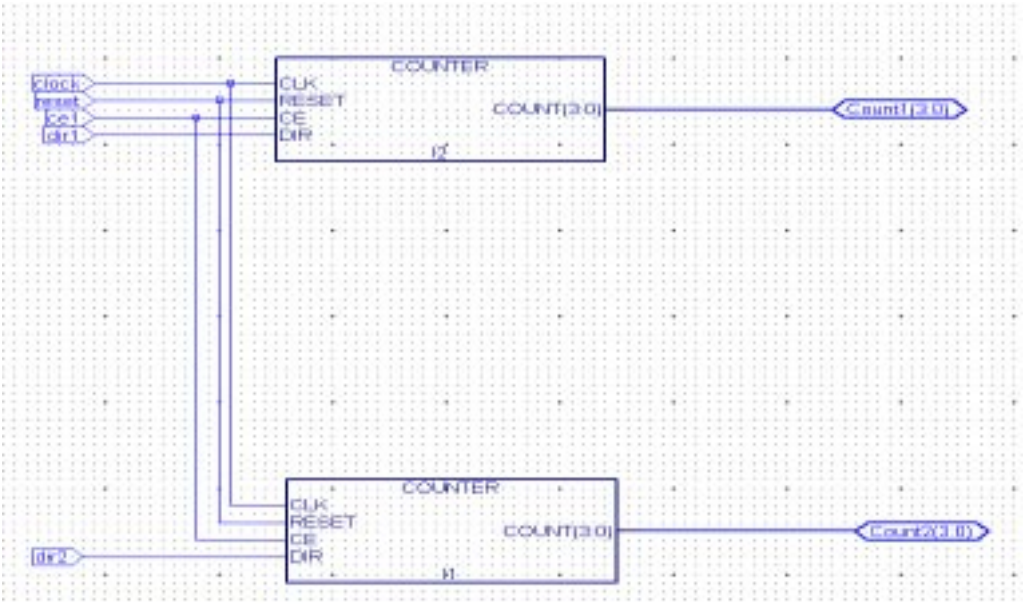


Figure 4-18 Completed Schematic

4. Exit the schematic editor. When asked to save changes, click Yes.

Note The Project Navigator automatically recognizes the design hierarchy and moves top.sch to the top of the design tree with counter.vhd listed as a sub module.

Design Implementation

Implement the design and use the Floorplanner to view the Placed and Routed design as follows:

1. Select top (top.sch) in the Source window.
2. Double-click Implement Design in the Process window.

   Note This will run all the processes (synthesis through Place-and Route) required to view the implemented design in the Floorplanner.
3. When implementation is finished, double click the Floorplanner which is located underneath Launch Tools in the Process window.
4. In the Floorplanner, open the file placed design “top.ngd” by performing the following steps.

   a) Select **File** → **Open**.

   b) Select the top.ngd file and click **Open**.

   c) In then New Floorplan dialog box, select **OK**. The top.fnf Placement window displays the design and its connections.
5. In the top.fnf Design Hierarchy window, select “top” (13 IOBs, 8 FGs, 8 CYS, 8 DFFs, 1 BUFG) and then click the Zoom to Selected button (the last icon on the far right of the toolbar) in the Floorplanner toolbar to zoom in.

   
6. Verify that all the I/Os are accounted for by holding the cursor over each of the pads and reading the pad name in the lower left corner of the Floorplanner window. The placement should look like the following figure.

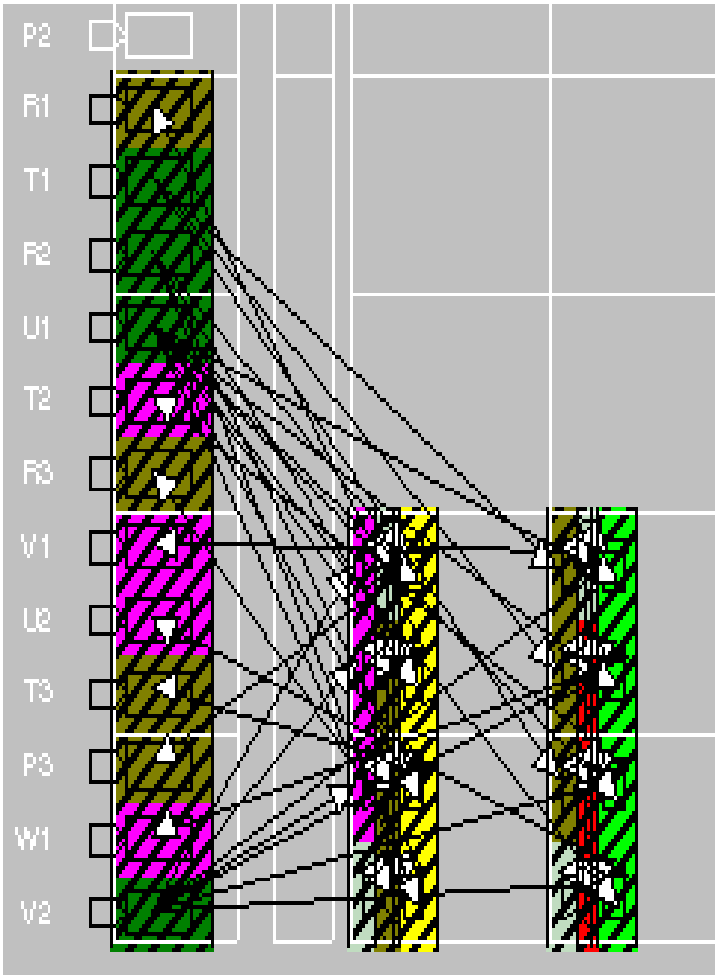


Figure 4-19 I/O Connections in Floorplanner

Timing Simulation

This section explains how to create a test bench using the HDL Bencher and then simulate the top-level design (top.sch) using ModelSim Xilinx Edition (MXE).

Before you can perform the timing simulation in this section, you *must* register the HDL Bencher and have a licensed version of MXE. For instructions on registering the Bencher and licensing MXE, refer to the "Installing Software" section of the "Setting Up the Tools" chapter for details.

1. Select top (top.sch) in the Source window.
2. Double-click Launch HDL Bencher Tool from the Process window.
3. Click **OK** to use the default timing constraints for the test bench. Your screen should look like this:

Figure 4-20 HDL Bencher Default Values

Note The blue areas are for entering input stimulus and the yellow areas are for entering expected response.

4. Initialize the counter as follows:
   a) Click the reset cell just to the right of clock cycle 1 until it is set to high.
   b) Click the reset cell just to the right of clock cycle 2 until it is set low.

c) Click the ce1 cell just to the right of clock cycle 3 until it is set high and enable the counter.

d) Click the dir1 cell just to the right of clock cycle 2 until it is set high.

e) Click the dir2 cell just to the right of clock cycle 1 until it is set high.

f) Click the dir2 cell just to the right of clock cycle 2 until it is set low.

5. Enter the expected response as follows:

a) Click the yellow COUNT1[3:0] cell in clock cycle 2 and click the Pattern button to launch the Pattern Wizard.

b) Set the pattern wizard parameters so that the expected output counts from 0 to 7 as follows:

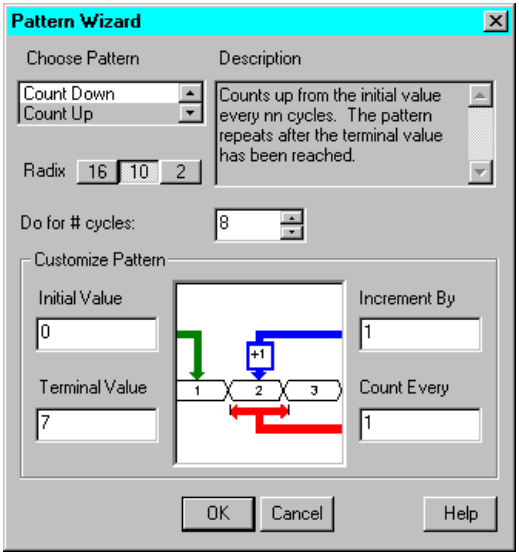


Figure 4-21 Pattern Wizard Settings

c) Click **OK** in the Pattern Wizard dialog box.

d) Click the yellow COUNT2[3:0] cell in clock cycle 2 and enter a 0 (zero) and press Enter.

e) Click the yellow COUNT2[3:0] cell in clock cycle 3 and click the Pattern button to launch the Pattern Wizard.

f) Set the pattern wizard parameters so that the expected output counts from 15 to 9 as follows:

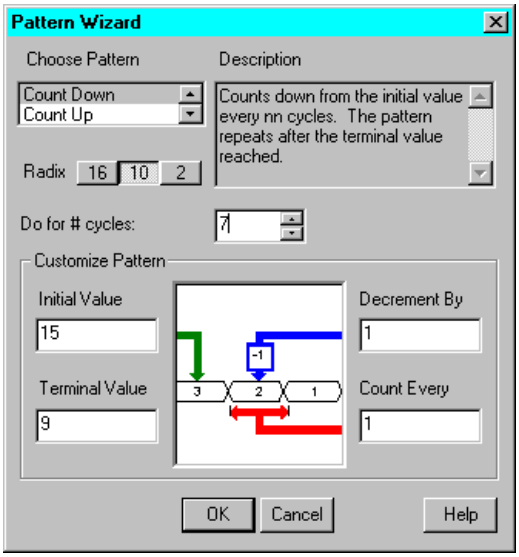


Figure 4-22 Pattern Wizard Settings

g) Click **OK** in the Pattern Wizard dialog box.

The Test Bench waveform should look like the following:

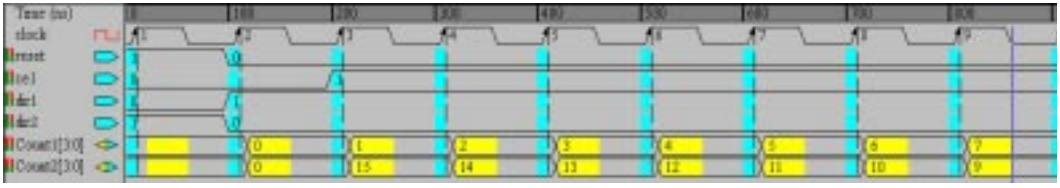


Figure 4-23 Testbench Waveform

6. Click the Export Testbench icon to create the test bench.

7. Close the Edit Test Bench window.
8. Save the test bench waveform by clicking the Save Waveform button in the toolbar.

9. Maximize the Project Navigator.
10. Add the test bench to the project by selecting **`Project`** → **`Add Source...`**.
11. Select the test bench file TOP_TB.VHD and click **`Open`**.
12. Select VHDL Test Bench as the source type and click **`OK`**.
13. Select "top" as the source to associate the test bench with and click **`OK`**.

    The Project Navigator Source window should look like the following:

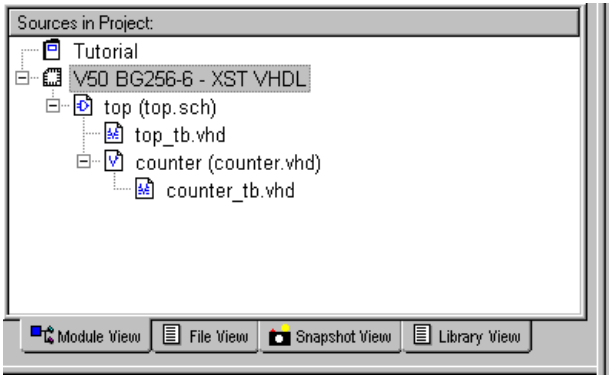


Figure 4-24 Test Bench File in Source Window

14. Select top_tb.vhd in the Project Navigator Source window and double click Simulate Post-Route VHDL Model in the process window. This will bring up the ModelSim screen.
15. Click Run ModelSim if the simulation does not run automatically.
16. Zoom out in the wave window until the waveform looks like the following:

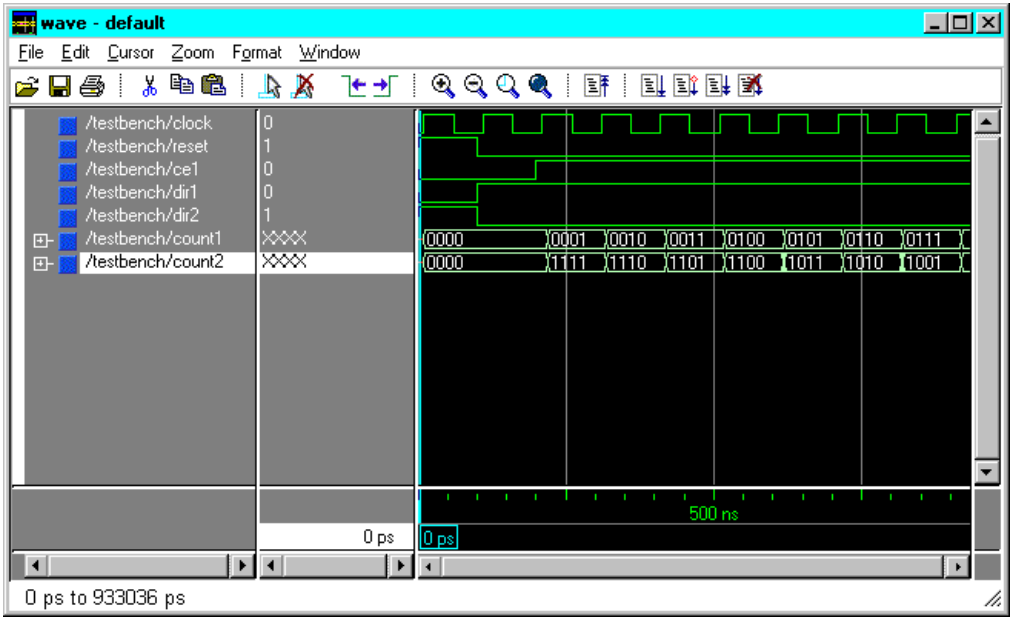


Figure 4-25 Timing Simulation Waveform

17. Verify that the timing simulation passes a 10ns clock to out requirement.

Glossary

ABEL

ABEL is a high-level language (HDL) and compilation system produced by Data I/O Corporation.

architecture

Architecture is the common logic structure of a family of programmable integrated circuits. The same architecture can be realized in different manufacturing processes. Examples of Xilinx architectures are the XC4000, Virtex, and XC9500 devices.

attributes

Attributes are instructions placed on symbols or nets in an FPGA schematic to indicate their placement, implementation, naming, direction, or other properties.

back-annotation

Back-annotation is the translation of a routed or fitted design to a timing simulation netlist.

black box Instantiation

Instantiation where the synthesizer is not given the architecture or modules.

CLB

The Configurable Logic Block (CLB) constitutes the basic FPGA cell. It includes two 16-bit function generators (F or G), one 8-bit function generator (H), two registers (flip-flops or latches), and reprogrammable routing controls (multiplexers).
CLBs are used to implement macros and other designed functions. They provide the physical support for an implemented and downloaded design. CLBs have inputs on each side, and this versatility makes them flexible for the mapping and partitioning of logic.

constraints

Constraints are specifications for the implementation process. There are several categories of constraints: routing, timing, area, mapping, and placement constraints.

Using constraints, you can force the placement of logic (macros) in CLBs, the location of CLBs on the chip, and the maximum delay between flip-flops. CLBs are arranged in columns and rows on the FPGA device. The goal is to place logic in columns on the device to attain the best possible placement from the standpoint of both performance and space.

constraints editor

A GUI tool that you can use to enter design constraints. In ISE, there are two constraint editors. The Express editor is available only in the FPGA Express product configuration. The Xilinx Constraints Editor is integrated with the Design Implementation tools and available in all product configurations.

constraint file

A constraint file specifies constraints (location and path delay) information in a textual form. An alternate method is to place constraints on a schematic.

CORE Generator tool

A software tool for generating and delivering parameterizable cores optimized for FPGAs. The library includes cores as complex as DSP filters and multipliers and cores as simple as delay elements. You can use these cores as building blocks in order to complete your designs more quickly.

CPLD

Complex Programmable Logic Device (CPLD) is an erasable programmable logic device that can be programmed with a schematic or a behavioral design. CPLDs constitute a type of complex PLD based on EPROM or EEPROM technology. They are characterized by an architecture offering high speed, predictable timing, and simple software.

The basic CPLD cell is called a macrocell, which is the CPLD implementation of a CLB. It is composed of AND gate arrays and is surrounded by the interconnect area.

CPLDs consume more power than FPGA devices, are based on a different architecture, and are primarily used to support behavioral designs and to implement complex counters, complex state machines, arithmetic operations, wide inputs, and PAL crunchers.

CPLD fitter

The CPLD Fitter implements designs for the XC9500 devices.

design entry tools

A set of tools accessible from the Project Navigator. These tools include the Schematic Editor, Symbol Editor, StateCAD and HDL Editor.

design implementation tools

A set of tools that comprise the mainstream programs offered in the Xilinx design implementation tools. The tools are NGDBuild, MAP, PAR, NGDAnno, TRCE, and all the NGD2 translator tools. The GUI-based implementation tools can be run by double clicking Implement Design from the Process window.

EDIF

Electronic Data Interchange Format. An industry standard for netlists.

effort level

Effort level refers to how hard the Xilinx tools try to place a design. The effort level settings are as follows.

- High, which provides the highest quality placement but requires the longest execution time. Use high effort on designs that do not route or do not meet your performance requirements.
- Medium, which is the default effort level. It provides the best trade-off between execution time and high quality placement for most designs.
- Low, which provides the fastest execution time and adequate placement results for prototyping of simple, easy-to-route designs. Low effort is useful if you are exploring a large design space and only need estimates of final performance.

fanout

Fanout is the maximum number of specified unit loads that a specified output can drive.

fitter

The fitter is the software that maps a PLD logic description into the target CPLD.

floorplanning (Edit Layout)

Floorplanning is the process of choosing the best grouping and connectivity of logic in a design.

It is also the process of manually placing blocks of logic in an FPGA where the goal is to increase density, routability, or performance.

FPGA

Field Programmable Gate Array (FPGA) is a class of integrated circuits pioneered by Xilinx in which the logic function is defined by the customer using Xilinx development system software after the IC has been manufactured and delivered to the end user. Gate arrays are another type of IC whose logic is defined during the manufacturing process. Xilinx supplies RAM-based FPGA devices.

FPGA applications include fast counters, fast pipelined designs, register intensive designs, and battery powered multi-level logic.

FSM

Finite State Machine

functional simulation

Functional simulation is the process of identifying logic errors in your design before it is implemented in a Xilinx device. Because timing information for the design is not available, the simulator tests the logic in the design using unit delays. Functional simulation is usually performed at the early stages of the design process. For ISE, functional simulation is performed prior to synthesis.

HDL

Hardware Description Language. A language that describes circuits in textual code. The two most widely accepted HDLs are VHDL and Verilog.

An HDL, or hardware description language, describes designs in a technology-independent manner using a high level of abstraction.

HDL Editor

Foundation ISE editor for ABEL, Verilog, and VHDL. The HDL Editor also provides a syntax checker, and language templates.

hierarchical design

Designs that are broken into multiple levels to clarify the design's function or permit easy reuse of functional blocks

implementation

Implementation is the mapping, placement, and routing of a design. A phase in the design process during which the design is placed and routed. (For CPLDs, the design is fitted.)

instantiation

Incorporating a macro or module into a top-level design. The instantiated module can be a LogiBLOX module, CORE-generated module, VHDL module, Verilog module, schematic module, state machine, or netlist.

I/O

Input/Output

IOB

Input/Output Block

LogiBLOX

Xilinx design tool for creating high-level modules such as counters, shift registers, and multiplexers.

macro

A macro is a component made of nets and primitives, flip-flops or latches, that implements high-level functions, such as adders, subtractors, and dividers. Soft macros and RPMs are types of macros.

A macro can be unplaced, partially placed, or fully placed; it can also be unrouted, partially routed, or fully routed. See also “physical macro.”

mapping

Mapping is the process of assigning a design’s logic elements to the specific physical elements that actually implement logic functions in a device.

MRP file

An MRP (mapping report) file is an output of the MAP run. It is an ASCII file containing information about the MAP run. The information in this file contains DRC (Design Rule Checking) warnings and messages, mapper warnings and messages, design information, schematic attributes, removed logic, expanded logic, signal cross references, symbol cross references, physical design errors and warnings, and a design summary.

NCD file

An NCD (netlist circuit description) file is the output design file from the MAP program, LCA2NCD, PAR, or FPGA Editor. It is a flat physical design database correlated to the physical side of the NGD in order to provide coupling back to the user's original design. The NCD file is an input file to MAP, PAR, TRCE, BitGen, and NGDAnno.

net

A net is a logical connection between two or more symbol instance pins. After routing, the abstract concept of a net is transformed to a physical connection called a wire.

A net is an electrical connection between components or nets. It can also be a connection from a single component. It is the same as a wire or a signal.

netlist

A netlist is a text description of the circuit connectivity. It is basically a list of connectors, a list of instances, and, for each instance, a list of the signals connected to the instance terminals. In addition, the netlist contains attribute information.

NGDBuild

The NGDBuild program performs all the steps necessary to read a netlist file in XNF or EDIF format and create an NGD file describing the logical design. The GUI equivalent is called Translate.

NGD file

An NGD (native generic database) file is an output from the NGDBuild run. An NGD file contains a logical description of the design expressed both in terms of the hierarchy used when the design was first created and in terms of lower-level Xilinx primitives to which the hierarchy resolves.

NGM file

An NGM (native generic mapping) file is an output from the MAP run and contains mapping information for the design. The NGM file is an input file to the NGDAnno program.

optimization

Optimization is the process that decreases the area or increases the speed of a design. Foundation allows you to control optimization of a design on a module-by-module basis. This means that you have the ability to, for instance, optimize certain modules of your design for speed, some for area, and some for a balance of both.

PAR

Place and Route

path delay

A path delay is the time it takes for a signal to propagate through a path.

PCF file

The PCF file is the “Physical Constraints File” created by the MAP program. It is an ASCII file containing physical constraints created by the MAP program as well as physical constraints you enter. You can edit the PCF file from within the FPGA Editor.

pin

A pin can be a symbol pin or a package pin. A package pin is a physical connector on an integrated circuit package that carries signals into and out of an integrated circuit. A symbol pin, also referred to as an instance pin, is the connection point of an instance to a net.

project

A project is a design. Each project has its own directory in which all source files, intermediate data files, and resulting files are stored.

Project Navigator

The primary GUI for managing an ISE Project. Design entry, synthesis, simulation, implementation, and programming files can be launched from the Project Navigator.

PROM File Formatter

The PROM File Formatter is the program used to format one or more bitstreams into an MC86, TEKHEX, EXORmacs or HEX PROM file format.

route

The process of assigning logical nets to physical wire segments in the FPGA that interconnect logic cells.

route-through

A route that can pass through an occupied or an unoccupied CLB site is called a route-through. You can manually do a route-through in the FPGA Editor. Route-throughs provide you with routing resources that would otherwise be unavailable.

static timing analysis

A static timing analysis is a point-to-point delay analysis of a design network.

static timing analyzer

A static timing analyzer is a tool that analyzes the timing of the design on the basis of its paths.

synthesis

The HDL design process in which each design module is elaborated and the design hierarchy is created and linked to form a unique design implementation. Synthesis starts from a high level of logic abstraction (typically Verilog or VHDL) and automatically creates a lower level of logic abstraction using a library containing primitives.

TRCE

TRCE (Timing Reporter and Circuit Evaluator) “trace” is a program that will automatically perform a static timing analysis on a design using the specified timing constraints. The input to TRCE is an NCD file and, optionally, a PCF file. The output from TRCE is an ASCII timing report which indicates how well the timing constraints for your design have been met.

TWR file

A TWR (Timing Wizard Report) file is an output from the TRCE program. A TWR file contains a logical description of the design expressed both in terms of the hierarchy used when the design was first created and in terms of lower-level Xilinx primitives to which the hierarchy resolves.

UCF file

A UCF (User Constraints File) contains user-specified logical constraints.

verification

Verification is the process of reading back the configuration data of a device and comparing it to the original design to ensure that all of the design was correctly received by the device.

Verilog

An industry-standard HDL (IEEE Std 1364) originally developed by Cadence Design Systems, now maintained by OVI. Recognizable as a file with a .v extension.

Verilog is a commonly used Hardware Description Language (HDL) that can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate level. It is IEEE standard 1364-1995.

VHDL

VHSIC (VHSIC is an acronym for Very High-Speed Integrated Circuits) Hardware Description Language. An industry-standard (IEEE 1076.1) HDL. Recognizable as a file with a .vhd or .vhdl extension.

VHDL can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate level. It is IEEE standard 1076-1993.

A language that is capable of describing the concurrent and sequential behavior of a digital system with or without timing.

XST

A Foundation ISE tool that synthesizes HDL designs.

Index

L

M

N

O

P

R

S

T