

# ISE Quick Start Tutorial





"Xilinx" and the Xilinx logo shown above are registered trademarks of Xilinx, Inc. Any rights not expressly granted herein are reserved. CoolRunner, RocketChips, Rocket IP, Spartan, StateBENCH, StateCAD, Virtex, XACT, XC2064, XC3090, XC4005, and XC5210 are registered trademarks of Xilinx, Inc.



The shadow X shown above is a trademark of Xilinx, Inc.

ACE Controller, ACE Flash, A.K.A. Speed, Alliance Series, AllianceCORE, Benchner, ChipScope, Configurable Logic Cell, CORE Generator, CoreLINX, Dual Block, EZTag, Fast CLK, Fast CONNECT, Fast FLASH, FastMap, Fast Zero Power, Foundation, Gigabit Speeds...and Beyond!, HardWire, HDL Benchner, IRL, J Drive, JBits, LCA, LogiBLOX, Logic Cell, LogiCORE, LogicProfessor, MicroBlaze, MicroVia, MultiLINX, NanoBlaze, PicoBlaze, PLUSASM, PowerGuide, PowerMaze, QPro, Real-PCI, RocketIO, SelectIO, SelectRAM, SelectRAM+, Silicon Xpresso, Smartguide, Smart-IP, SmartSearch, SMARTswitch, System ACE, Testbench In A Minute, TrueMap, UIM, VectorMaze, VersaBlock, VersaRing, Virtex-II Pro, Virtex-II EasyPath, Wave Table, WebFITTER, WebPACK, WebPOWERED, XABEL, XACT-Floorplanner, XACT-Performance, XACTstep Advanced, XACTstep Foundry, XAM, XAPP, X-BLOX +, XC designated products, XChecker, XDM, XEPLD, Xilinx Foundation Series, Xilinx XDTV, Xinfo, XSI, XtremeDSP and ZERO+ are trademarks of Xilinx, Inc.

The Programmable Logic Company is a service mark of Xilinx, Inc.

All other trademarks are the property of their respective owners.

Xilinx, Inc. does not assume any liability arising out of the application or use of any product described or shown herein; nor does it convey any license under its patents, copyrights, or maskwork rights or any rights of others. Xilinx, Inc. reserves the right to make changes, at any time, in order to improve reliability, function or design and to supply the best product possible. Xilinx, Inc. will not assume responsibility for the use of any circuitry described herein other than circuitry entirely embodied in its products. Xilinx provides any design, code, or information shown or described herein "as is." By providing the design, code, or information as one possible implementation of a feature, application, or standard, Xilinx makes no representation that such implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of any such implementation, including but not limited to any warranties or representations that the implementation is free from claims of infringement, as well as any implied warranties of merchantability or fitness for a particular purpose. Xilinx, Inc. devices and products are protected under U.S. Patents. Other U.S. and foreign patents pending. Xilinx, Inc. does not represent that devices shown or products described herein are free from patent infringement or from any other third party right. Xilinx, Inc. assumes no obligation to correct any errors contained herein or to advise any user of this text of any correction if such be made. Xilinx, Inc. will not assume any liability for the accuracy or correctness of any engineering or software support or assistance provided to a user.

Xilinx products are not intended for use in life support appliances, devices, or systems. Use of a Xilinx product in such applications without the written consent of the appropriate Xilinx officer is prohibited.

The contents of this manual are owned and copyrighted by Xilinx. Copyright 1994-2003 Xilinx, Inc. All Rights Reserved. Except as stated herein, none of the material may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of any material contained in this manual may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.



## About This Tutorial

---

The ISE Quick Start Tutorial is a hands-on learning tool for new users of the ISE software and for users who wish to refresh their knowledge of the software. This tutorial is current for ISE 6.x. The tutorial demonstrates basic set-up and design methods available in the PC version of the ISE software. By the end of the tutorial, you will have a greater understanding of how to implement your own design flow using the ISE software.

In the ISE Quick Start Tutorial, you will create a new project called Tutorial, in which you will design a 4-bit counter module, simulate and implement the design, and view the results.

Following the ISE Quick Start Tutorial, an appendix, EDIF Design, demonstrates how to implement an existing netlist using the ISE software.

## Manual Contents

This manual contains the following chapters:

- “VHDL and Schematic Design Flow,” demonstrates how to use the VHDL and schematic design entry tools, how to perform behavioral and timing simulation, and how to implement a design.
- Appendix A, “EDIF Design Flow,” explains how to implement a design in ISE from an EDIF source file.

## Additional Resources

For additional information, go to <http://support.xilinx.com>. The following table lists some of the resources you can access from this website. 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 <a href="http://support.xilinx.com/support/techsup/tutorials/index.htm">http://support.xilinx.com/support/techsup/tutorials/index.htm</a>
Answer Browser	Database of Xilinx solution records <a href="http://support.xilinx.com/xlnx/xil_ans_browser.jsp">http://support.xilinx.com/xlnx/xil_ans_browser.jsp</a>
Application Notes	Descriptions of device-specific design techniques and approaches <a href="http://support.xilinx.com/apps/appsweb.htm">http://support.xilinx.com/apps/appsweb.htm</a>

Resource	Description/URL
Data Book	Pages from <i>The Programmable Logic Data Book</i> , which contains device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging <a href="http://support.xilinx.com/partinfo/databook.htm">http://support.xilinx.com/partinfo/databook.htm</a>
Problem Solvers	Interactive tools that allow you to troubleshoot your design issues <a href="http://support.xilinx.com/support/troubleshoot/psolvers.htm">http://support.xilinx.com/support/troubleshoot/psolvers.htm</a>
Tech Tips	Latest news, design tips, and patch information for the Xilinx design environment <a href="http://www.support.xilinx.com/xlnx/xil_tt_home.jsp">http://www.support.xilinx.com/xlnx/xil_tt_home.jsp</a>

## Conventions

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

### Typographical

The following typographical conventions are used in this document:

Convention	Meaning or Use	Example
Courier font	Messages, prompts, and program files that the system displays	speed grade: - 100
<b>Courier bold</b>	Literal commands that you enter in a syntactical statement	<b>ngdbuild</b> <i>design_name</i>
<b>Helvetica bold</b>	Commands that you select from a menu	<b>File → Open</b>
	Keyboard shortcuts	<b>Ctrl+C</b>
<i>Italic font</i>	Variables in a syntax statement for which you must supply values	<b>ngdbuild</b> <i>design_name</i>
	References to other manuals	See the <i>Development System Reference Guide</i> for more information.
	Emphasis in text	If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <i>not</i> connected.
Square brackets [ ]	An optional entry or parameter. However, in bus specifications, such as <b>bus[ 7:0 ]</b> , they are required.	<b>ngdbuild</b> [ <i>option_name</i> ] <i>design_name</i>
Braces { }	A list of items from which you must choose one or more	<b>lowpwr</b> = { <b>on</b>   <b>off</b> }

Convention	Meaning or Use	Example
Vertical bar	Separates items in a list of choices	<code>lowpwr = {on   off}</code>
Vertical ellipsis . . .	Repetitive material that has been omitted	IOB #1: Name = QOUT' IOB #2: Name = CLKIN' . . .
Horizontal ellipsis ...	Repetitive material that has been omitted	<code>allow block block_name loc1 loc2 ... locn;</code>

## Online Document

The following conventions are used in this document:

Convention	Meaning or Use	Example
Blue text	Cross-reference link to a location in the current file or in another file in the current document	See the section “ <a href="#">Additional Resources</a> ” for details. Refer to “ <a href="#">Title Formats</a> ” in <a href="#">Chapter 1</a> for details.
Red text	Cross-reference link to a location in another document	See <a href="#">Figure 2-5</a> in the <i>Virtex-II Handbook</i> .
<a href="#">Blue, underlined text</a>	Hyperlink to a website (URL)	Go to <a href="http://www.xilinx.com">http://www.xilinx.com</a> for the latest speed files.





# Table of Contents

---

## Preface: About This Tutorial

Manual Contents .....	5
Additional Resources .....	5
Conventions .....	6
Typographical .....	6
Online Document .....	7

## : VHDL and Schematic Design Flow

Tutorial Overview .....	11
Getting Started .....	11
Software Requirements .....	12
Starting the ISE Software .....	12
Stopping and Restarting your Session .....	12
Accessing Help .....	12
Design Entry (VHDL) .....	12
Creating a New Project .....	13
Modifying Counter Module with Counter Template .....	15
Simulating the Behavioral Model .....	17
Creating a Test Bench Waveform Source .....	18
Initializing Counter Inputs .....	18
Generating the Expected Simulation Output Values .....	19
Simulating with ModelSim .....	20
Behavioral Simulation .....	20
Post-Place and Route Simulation .....	21
Design Entry (Schematic) .....	22
Creating a Schematic Symbol for the VHDL Module .....	22
Creating a New Top-Level Schematic .....	23
Instantiating VHDL Modules .....	23
Wiring the Schematic .....	24
Adding Net Names to Wires .....	25
Creating Buses .....	26
Adding I/O Markers .....	27
Design Implementation .....	28
Running Implement Design .....	29
Viewing the Design in Floorplanner .....	29
Simulating the Top-level Design .....	30
Creating a Test Bench Waveform Source .....	30
Initializing Counter Inputs .....	31
Generating the Expected Responses .....	31
Post-place and Route Simulation .....	34

## Appendix A: EDIF Design Flow

EDIF Overview .....	35
Design Entry .....	35

Creating a New Project .....	35
<b>Design Implementation</b> .....	37
Running Implement Design .....	37
Viewing the Design in FPGA Editor .....	38
<b>Index</b> .....	41



# VHDL and Schematic Design Flow

---

The ISE Quick Start Tutorial describes and demonstrates how to use the VHDL and schematic design entry tools, how to perform behavioral and timing simulation, and how to implement a design.

You can use both HDL and Schematic sources in an HDL flow, and in a Schematic flow.

**Note:** This tutorial is designed for ISE 6.x, PC version.

This tutorial contains the following sections.

- “Tutorial Overview”
- “Getting Started”
- “Design Entry (VHDL)”
- “Simulating the Behavioral Model”
- “Design Entry (Schematic)”
- “Design Implementation”
- “Simulating the Top-level Design”

To learn how to import your own netlist into ISE and view the design, see “EDIF Design Flow”

For an in-depth explanation of the ISE design tools, see the ISE In-Depth Tutorial on the Xilinx web site (<http://www.support.xilinx.com/support/techsup/tutorials/>).

## Tutorial Overview

Once you have completed the tutorial you will know how to do the following:

- Create a project with a Virtex device.
- Create a VHDL module for a 4-bit counter using the ISE Language Templates.
- Create a test bench waveform source used to simulate the behavior of the 4-bit counter.
- Create a top-level schematic design.
- Instantiate two VHDL counter modules into the top-level schematic design.
- Wire modules together and add net names, buses, and I/O markers.
- Apply timing constraints, input initialization and response constraints to the 4-bit counter waveform and to the top-level schematic waveform.
- Perform behavioral and timing simulations on the 4-bit counter and timing simulation on the top-level schematic design.
- View the placed and routed design in the Floorplanner.

## Getting Started

This section describes the software requirements for this tutorial, how to start up the PC version of the software and how to access online help resources.

## Software Requirements

To follow along with this tutorial, you will need the following software installed:

- ISE 6.x
- A licensed ModelSim simulator that supports VHDL simulation.

For more information about installing Xilinx software, see *ISE Release Notes and Installation Guide*.

## Starting the ISE Software

For PC users, start ISE from the Start menu by selecting **Start** → **Programs** → **Xilinx ISE 6.x** → **Project Navigator**.

**Note:** Your start-up path is set during the installation process and may differ from the one above.

## Stopping and Restarting your Session

At any point during this tutorial you can stop your session and continue at a later time.

To stop the session:

1. Save all source files open in other applications.
2. Exit the software (ISE and other application).  
When you exit ISE, the project file is automatically saved with the most recent changes made.

To restart your session:

1. Start ISE.  
ISE displays the content of your project with the last saved changes.
2. To view or to continue to edit a source file, double-click a file to display the contents of the file.

## Accessing Help

At any time during the tutorial, you can access online help for further information on a variety of topics and procedures in the ISE software.

- Press F1 to view the help for the specific tool or function you are currently using. For example, when you press F1 while in Floorplanner, Floorplanner online help is displayed.
- Design flow-based help called *ISE Help* is accessible from the Help menu in Project Navigator. This help package contains information about creating and maintaining your complete design flow in ISE.

## Design Entry (VHDL)

In this section, you will create a 4-bit counter module using a top-level schematic as a block editor. The design contains HDL counter modules created with examples from the Language Templates. To begin, create a new project and counter module, then modify the counter module with the counter template.

## Creating a New Project

A project in ISE is a collection of all files necessary to create and download a design to the selected device. To create a new project for this tutorial:

1. Select **File** → **New Project**.
2. In the New Project Wizard dialog box, type the desired location in the Project Location field, or browse to the directory under which you want to create your new project directory using the browse button next to the Project Location field.
3. Enter 'Tutorial' in the Project Name field.

When you enter 'Tutorial' in the Project Name field, a Tutorial subdirectory is automatically created in the directory path in the Project Location field. For example, for the directory path C:\My\_Projects, entering the Project Name 'Tutorial' modifies the path as C:\My\_Projects\Tutorial.

4. Use the pull-down arrow to select Schematic from the Top-Level Module Type field. Click in the field to access the pull-down list.

**Note:** You can apply the fundamentals learned from this tutorial to either an HDL or schematic design containing both schematic and/or HDL sources. In this example, you will use a top-level schematic module and lower-level HDL modules.

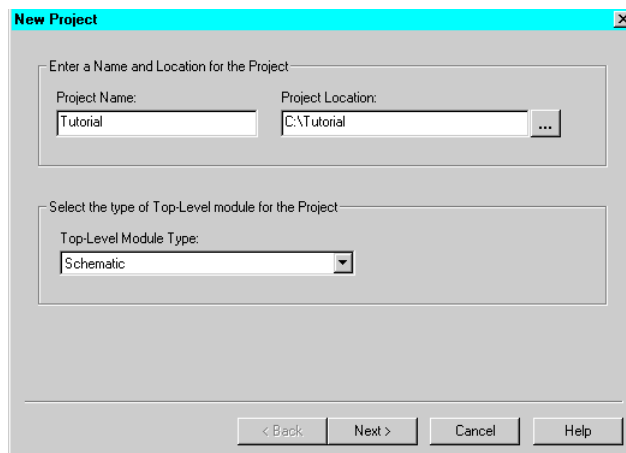


Figure 1: New Project Wizard Dialog Box

5. Click **Next**

6. In the New Project Wizard Device and Design Flow dialog box, use the pull-down arrow to select the Value for each Property Name. Click in the field to access the pull-down list.

Change the values as follows:

- ◆ Device Family: Virtex
- ◆ Device: xcv50
- ◆ Package: bg 256
- ◆ Speed Grade: -6
- ◆ Synthesis Tool: XST (VHDL/Verilog)
- ◆ Simulator: Modelsim
- ◆ Generated Simulation Language: VHDL

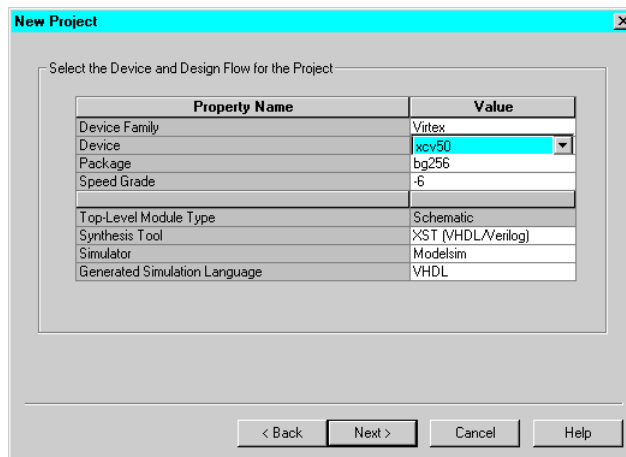


Figure 2: New Project Wizard Device and Design Flow Dialog Box

7. Click **Next**.

Next, create a VHDL module for a counter. To create a counter module:

8. Click **New Source** in the New Project Wizard Create a New Source Dialog box to add one new source to your project.
9. In the New Source dialog box, select **VHDL Module** as the source type.
10. Type in the file name 'counter'.
11. Verify that the "Add to Project" checkbox is selected.
12. Click **Next**.
13. Click **Next** in the Define VHDL Source dialog box.
 

**Note:** You have the option of defining and adding ports from this dialog box. In this example, you will use pre-defined ports supplied in the Language Template.
14. Click **Finish** to complete the new source file template.
15. Click **Next** in the New Project Wizard Create a New Source dialog box.
16. Click **Next** in the New Project Wizard Add Existing Sources dialog box.
17. Click **Finish** in the New Project Wizard Summary dialog box.

ISE creates and displays the new project in the Sources in Project window, and opens the **counter.vhd** file in ISE Text Editor.

Counter.vhd, which is displayed in the ISE Text Editor window, contains the library declaration and use statements along with the empty entity and architecture pair for the counter you have just created. The file (when correctly placed in your design) is all you need to create a counter module. ISE Text Editor is designed for editing HDL source files.

## Modifying Counter Module with Counter Template

To complete the counter module, insert port declarations and the behavioral code for the VHDL counter from the ISE Language Templates. The Language Templates contains many ABEL, Verilog and VHDL language templates for use in a design.

1. Open the Language Templates by selecting **Edit → Language Templates** or by clicking the Language Templates icon located on the far right on the toolbar.



Figure 3: Language Templates icon

**Note:** If the Language Template icon is not displayed in the toolbar, select **View → Editor Toolbar**.

**Note:** Undock the Language Templates window for a larger display using the  button.

2. In the Language Templates window, click the + sign next to **VHDL** to expand the list of templates, then click the + sign next to **Synthesis Templates**, and select **Counter**.

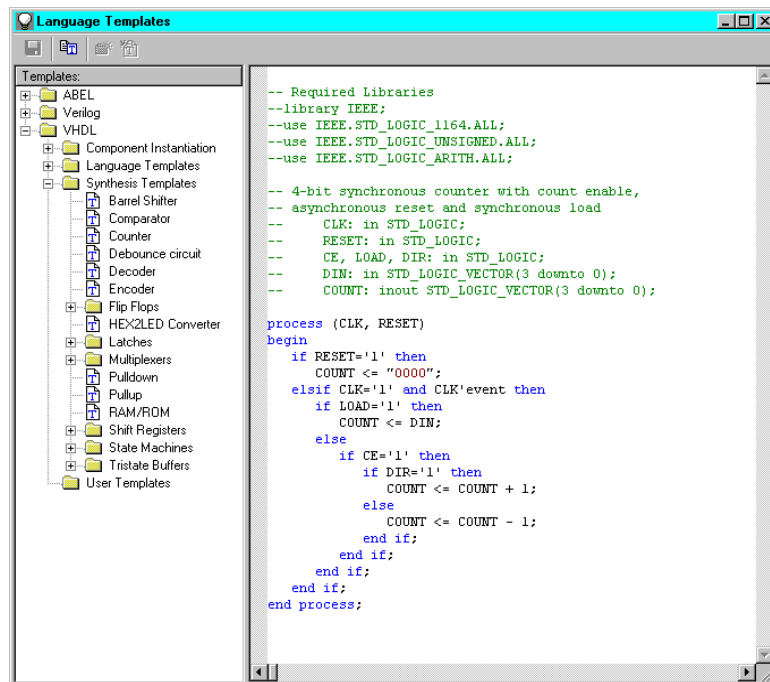


Figure 4: Counter Language Template

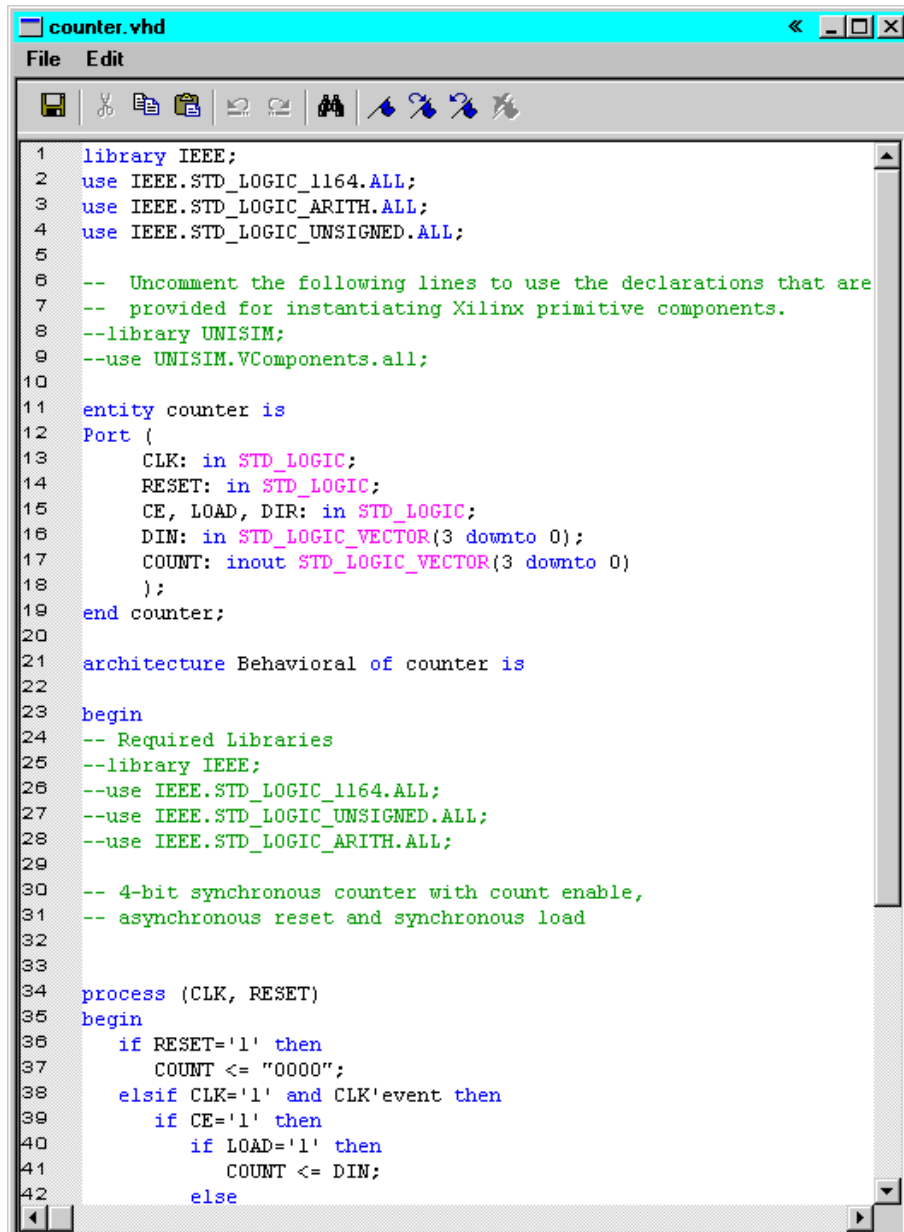
3. Copy the contents from the Counter template in the VHDL Synthesis Templates folder and paste them into counter.vhd between the begin and end behavioral statements.
4. Close the Language Templates window.

5. Type a carriage return after the `entity counter is` statement.
  6. On the following line, type an open port statement:  
`Port (`
  7. Cut the port definitions from the comment section of the counter.vhd file and paste them on the following line (this will be the port declaration of the counter entity in your file). The port definitions are the following lines:

```
-- 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);
```
  8. Uncomment the above port definitions in your counter.vhd file by removing the dashes from the beginning of each line.
  9. Remove the semicolon that follows the COUNT port definition and type a carriage return.  

```
COUNT: inout STD_LOGIC_VECTOR(3 downto 0)
```
  10. On the following line, close the port statement by typing:  
`);`
  11. Save counter.vhd by selecting **File** → **Save**.
- Your counter.vhd source should look like the following.





```

1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  -- Uncomment the following lines to use the declarations that are
7  -- provided for instantiating Xilinx primitive components.
8  --library UNISIM;
9  --use UNISIM.VComponents.all;
10
11  entity counter is
12  Port (
13      CLK: in STD_LOGIC;
14      RESET: in STD_LOGIC;
15      CE, LOAD, DIR: in STD_LOGIC;
16      DIN: in STD_LOGIC_VECTOR(3 downto 0);
17      COUNT: inout STD_LOGIC_VECTOR(3 downto 0)
18  );
19  end counter;
20
21  architecture Behavioral of counter is
22
23  begin
24      -- Required Libraries
25      --library IEEE;
26      --use IEEE.STD_LOGIC_1164.ALL;
27      --use IEEE.STD_LOGIC_UNSIGNED.ALL;
28      --use IEEE.STD_LOGIC_ARITH.ALL;
29
30      -- 4-bit synchronous counter with count enable,
31      -- asynchronous reset and synchronous load
32
33
34  process (CLK, RESET)
35  begin
36      if RESET='1' then
37          COUNT <= "0000";
38      elsif CLK='1' and CLK'event then
39          if CE='1' then
40              if LOAD='1' then
41                  COUNT <= DIN;
42              else

```

Figure 5: Modified Counter Module

12. Close the source file.

## Simulating the Behavioral Model

In this section, you will create a test bench waveform that defines the desired functionality for the counter module. This test bench waveform is then used in conjunction with a ModelSim simulator to verify that the counter design meets both behavioral and timing design requirements.

## Creating a Test Bench Waveform Source

First, create a test bench waveform in Project Navigator which you will modify in HDL Bench.

1. Select the counter (counter.vhd) in the Sources in Project window.
2. Select **Project** → **New Source**.
3. In the New Source dialog box, select the **Test Bench Waveform** source type.
4. Type the name 'counter\_tbw'.
5. Click **Next**.

**Note:** In other projects, you can associate your test bench waveform with other sources.

6. Click **Next**.
7. Click **Finish**.


HDL Bench is launched and ready for timing requirements to be entered.

You will now specify the timing parameters used during simulation. The Clock high time and Clock low time together define the clock period for which the design must operate. The Input setup time defines when inputs must be valid. The Output valid delay defines the time after active clock edge when the outputs must be valid.

For this tutorial, you will not change any of the default timing constraints. The default Initialize Timing settings are the following:

```
Clock high time: 50 ns
Clock low time: 50 ns
Input setup time: 10 ns
Output valid delay: 10 ns
Offset: 0 ns
```

8. Click **OK** to accept the default timing constraints.
9. HDL Bench opens within the Project Navigator framework.

**Note:** To use HDL Bench in a separate window, click on the  button to undock it from the framework.

Your test bench waveform should look like the following.

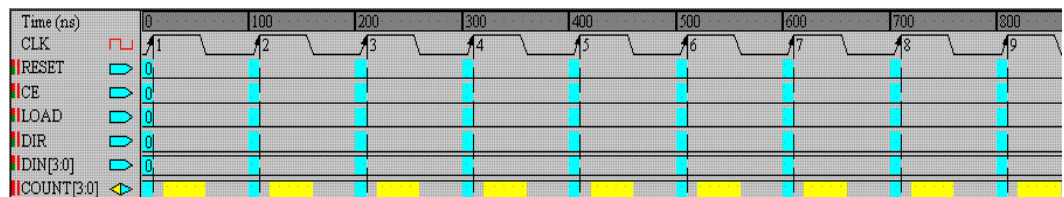


Figure 6: Test Bench waveform in HDL Bench

### Initializing Counter Inputs

**Note:** Before initializing the inputs, ensure that you have Radix set to Decimal. Radix is set to Decimal when the 10 icon in the HDL Bench toolbar is selected.

In the waveform in HDL Bench, initialize the counter inputs as follows. Verify your entries using the figure below.

**Note:** Enter the input stimulus in the blue area in each cell.

1. Click the RESET cell under CLK cycle 1 so that the cell is set high.

2. Click the RESET cell under CLK cycle 2 so the cell is set low.
  3. Click the CE cell under CLK cycle 3 so it is set high.
  4. Click the DIR cell under CLK cycle 2 so the cell is set high.
- Your test bench waveform should now look like the following.

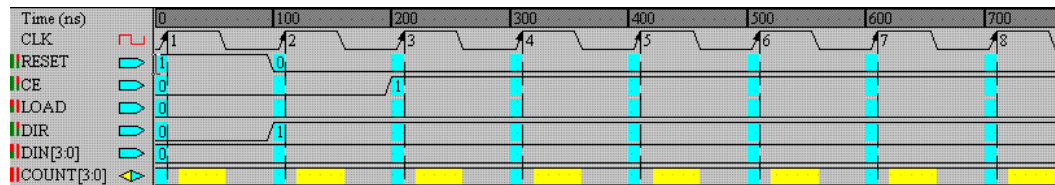


Figure 7: HDL Bencher Stimulus and Response Entries

5. Change the last test bench cycle to 11 using one of the following methods:
  - ♦ Place the cursor under clock cycle 11 and select **Set end of test bench** from the right-click menu.
  - or,
  - ♦ Click and drag the blue vertical line to clock cycle 11.

This extends the waveform 8 clock cycles past the assertion of CE high.
6. Save your test bench waveform by clicking the Save Waveform icon.



Figure 8: Save Waveform icon

7. Exit HDL Bencher.

The new test bench waveform source (counter\_tbw) is automatically added to the project.

## Generating the Expected Simulation Output Values

Now you can generate the expected outputs for the counter module based on the initialized inputs you have entered.

1. Select counter\_tbw.tbw in the Sources in Project window.
2. In the Processes for Source window, click the + beside **ModelSim Simulator** to expand the hierarchy.
3. Double-click **Generate Expected Simulation Results**.

This process runs a background simulation using the inputs specified, generating output values which are added to the test bench waveform.

Your test bench waveform, a counter beginning at cycle 3 and counting until cycle 11, should look like the following.

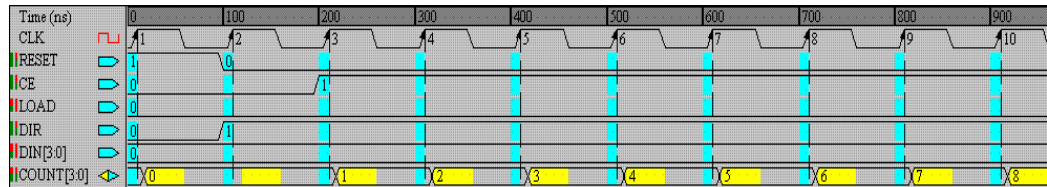


Figure 9: Generated Simulation Results

- Exit HDL Bencher without saving your waveform.

## Simulating with ModelSim

After viewing the expected results generated in HDL Bencher, you are now ready to run your simulation with ModelSim. Simulating the counter design verifies the logic and timing of the design. For this tutorial, you will run a behavioral simulation (also referred to as a functional simulation) and a post-place and route simulation.

### Behavioral Simulation

Run a behavioral simulation to verify the counter module's functionality.

- In Project Navigator, select counter\_tbw.tbw in the Sources in Project window.
- In the Processes for Source "counter\_tbw.tbw" window, double-click **Simulate Behavioral Model** found in the **ModelSim Simulator** hierarchy.

ModelSim is launched.

- For first-time users of ModelSim, a dialog box appears in which you:
  - Check the Do not show this dialog again option.
  - Click Run ModelSim.

This dialog box will not appear again unless you reinstall or reconfigure ModelSim.

Your simulation results are displayed in the ModelSim wave window. You may be required to maximize the ModelSim wave window from the task bar.

**Note:** ISE automates the simulation process by creating and launching a simulation macro file (an FDO file). Though not visible to the user, in this tutorial the counter\_tbw.fdo file performs the following functions:

- 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
- Runs the simulation for the time specified in HDL Bencher, 11 clock cycles (1100 ns).

4. In the Wave window, click **View** → **Zoom** → **Zoom Full** or click the Zoom Full icon in the toolbar.



Figure 10: Zoom Full icon

The output waveform should look like the following:.

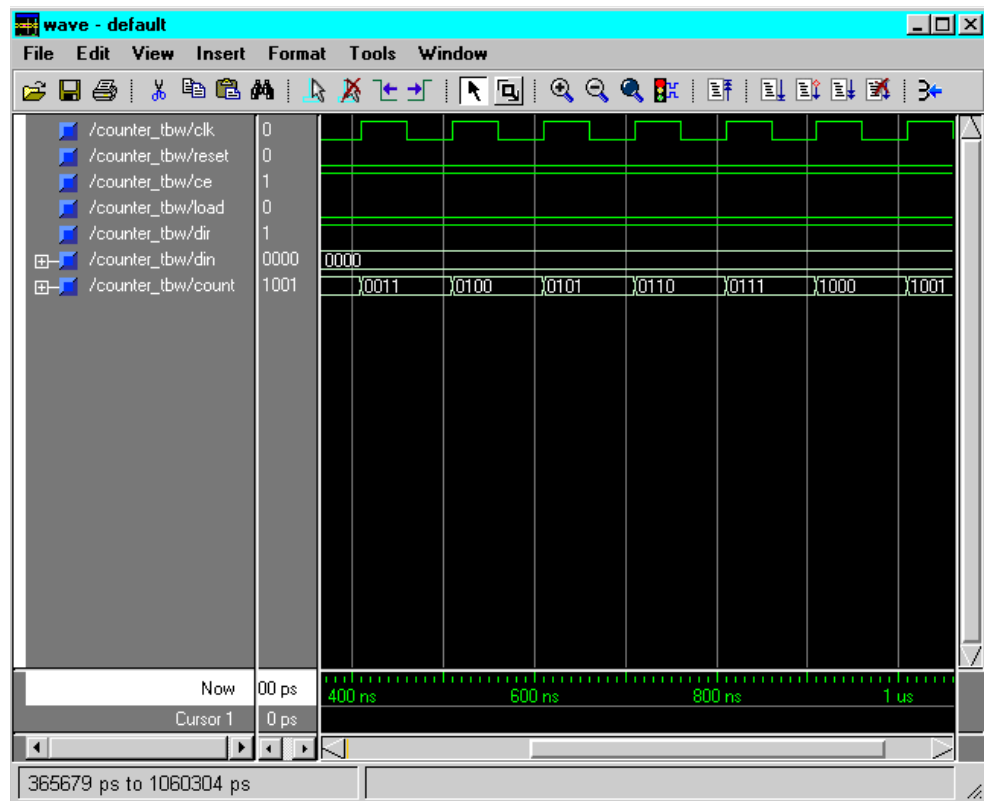


Figure 11: Behavioral Simulation Waveform

5. Exit ModelSim by closing the main ModelSim window.

## Post-Place and Route Simulation

The post-place and route simulation includes timing information for the targeted device. To perform post-place and route simulation on the counter module:

1. In Project Navigator, select counter\_tbw.tbw in the Sources in Project window.
2. In the Processes for Source window, double-click **Simulate Post-Place & Route VHDL Model** found in the **ModelSim Simulator** hierarchy.

**Note:** This will take the design through synthesis, place-and-route, implementation, and back-annotation.

ModelSim is launched.

3. Click **Zoom** → **Zoom Full** or click the Zoom Full icon in the toolbar.



Figure 12: **Zoom Full icon**

The output waveform looks like the following:.

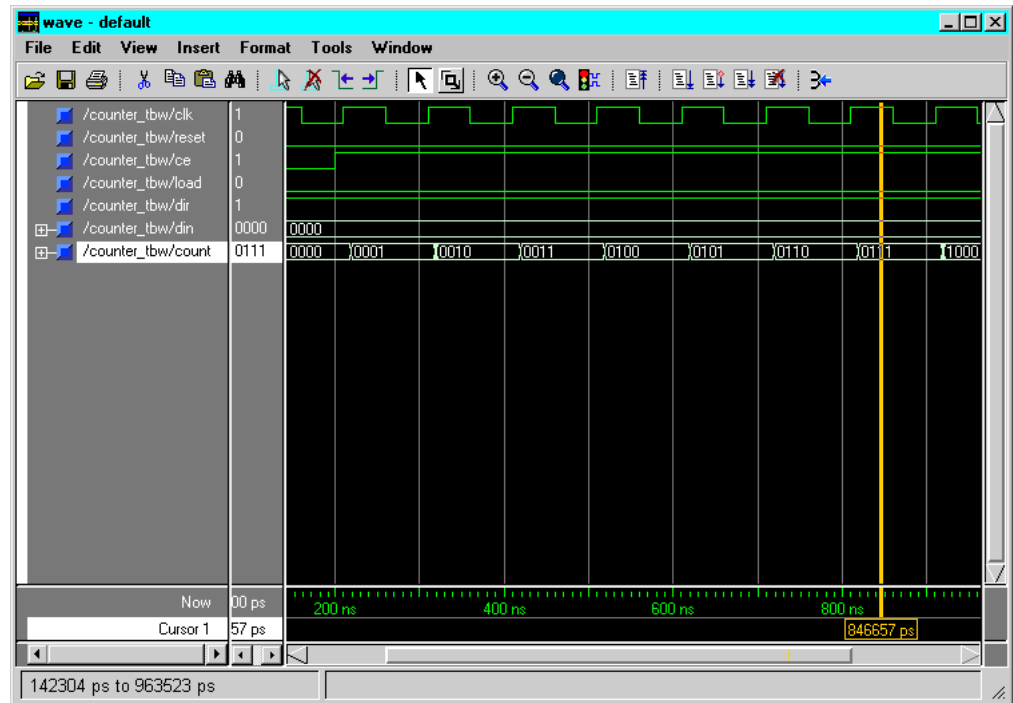


Figure 13: **Post-place and Route Simulation Waveform**

4. Exit ModelSim by closing the main ModelSim window.

## Design Entry (Schematic)

This section demonstrates how to create a top-level schematic that contains instantiations of the counter module, and describes how to wire together the modules, add net names and buses to the wires, and add I/O markers to show where signals enter or exit the schematic. This section of the tutorial introduces you to the Xilinx tool for creating and editing schematic diagrams: Engineering Capture System (ECS).

### Creating a Schematic Symbol for the VHDL Module

To create a schematic symbol for the VHDL module:

1. In Project Navigator, in the Sources in Project window, select your counter module, counter.vhd.

2. In the Processes for Source window, click the + sign beside **Design Entry Utilities** and double-click the **Create Schematic Symbol** process.

**Note:** This places a schematic component entitled 'counter' in the project library.

## Creating a New Top-Level Schematic

To create a new top-level schematic, in Project Navigator:

1. Select **Project** → **New Source**.
2. Select **Schematic** as the source type.
3. Type in the name 'top'.
4. Click **Next** and then click **Finish**.

ECS is launched and a blank sheet opens in an ECS schematic window. In ECS, you will create a schematic diagram from scratch.

## Instantiating VHDL Modules

With a blank sheet open in ECS, instantiate two counter module symbols in the top-level schematic.

1. Select **Add** → **Symbol** or click the Add Symbol icon in the Tools toolbar.



Figure 14: Add Symbol icon

2. Select **counter** from the Symbols list in the Symbol tab (to the left of the screen). Do not select any options from the Categories list.

**Caution!** Do not select the Counter from the Categories list. Be sure to select the counter symbol you created in your project (visible when your project or **All Symbols** is selected in the categories list).

3. Place two counters in the schematic. Click the left mouse button to place a counter on the schematic where the cursor sits.
4. Press **Esc** to exit Add Symbol mode and restore your cursor.

**Note:** Adjust your view using the Zoom option (**View** → **Zoom** → **In**) and the scroll bars in ECS.

Your schematic should look like the following diagram.

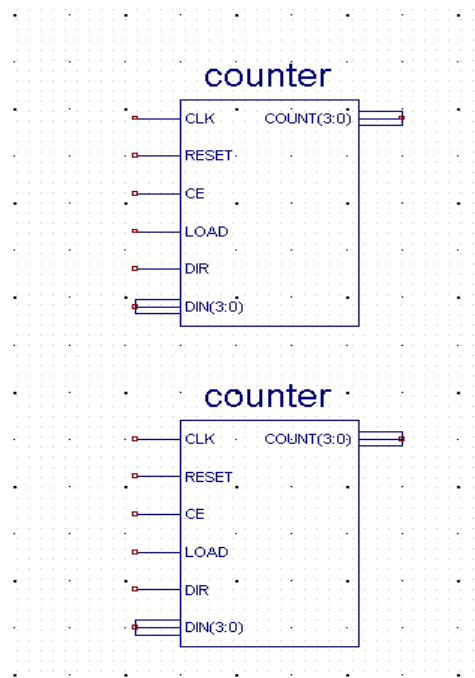


Figure 15: Instantiated VHDL Modules

## Wiring the Schematic

When wiring the schematic symbols, some wires interconnect the modules and others are extended and left hanging.

1. To activate the drawing tool, select **Add** → **Wire** or select the Add Wire icon from the Tools toolbar.



Figure 16: Add Wire icon

2. To add a hanging wire or to extend the wire for clk, reset, ce, load, dir, din(3:0) and count(3:0):
  - a. Click and hold the mouse button at the vertex of a pin on the first counter module.
  - b. Drag the mouse to extend the wire to the desired length.
  - c. Release the mouse button at the location you want the wire to terminate.

**Note:** Add hanging wires according to the diagram in [Figure 17](#).

3. To connect the wires clk, reset, ce and load from the second schematic symbol to the extended wires in the first schematic symbol:

- a. Click once at the vertex of a pin on the second counter module.
- b. Double-click anywhere on the destination wire of the first counter module.

**Note:** Connect the wires of the two counters according to the diagram in [Figure 17](#).



When finished wiring, press **Esc** to exit Add Wire mode.

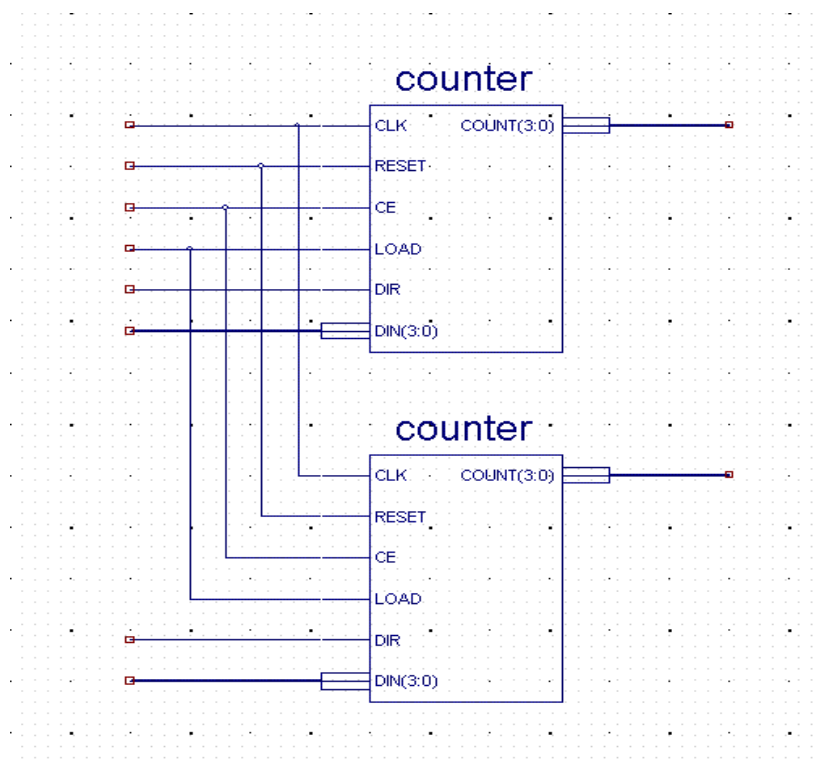


Figure 17: Interconnected Modules

## Adding Net Names to Wires

After wiring the schematic symbols, you are ready to add net names to the wires.

1. Select **Add** → **Net Name** or click the Add Net Name icon from the Tools toolbar.



Figure 18: Add Net Name icon

Next, add the following six net names to the schematic: clock, reset, ce, load, dir1, and dir2.

2. To create and place a net name for each hanging wire:
  - a. Type the net name in the text box in the Options tab, located to the left of the screen.
  - Note:** Leave the default options as Name the branch and Keep the name.
  - b. Place the cursor, which now displays the net name, at the end of the hanging wire.
  - c. Click the left mouse button.

With the six net names added, your schematic should look like the following diagram.

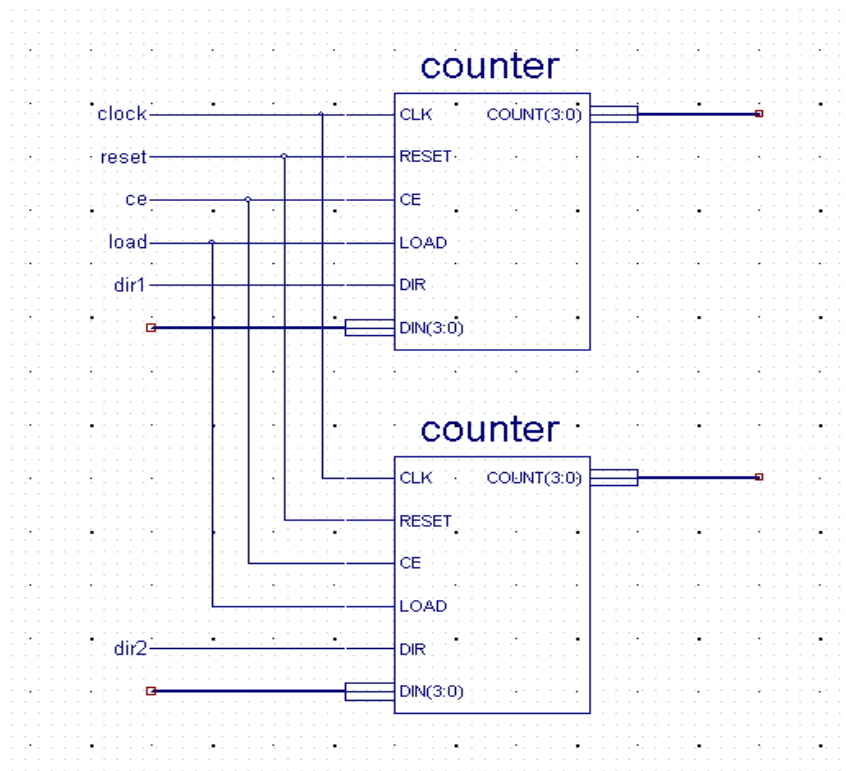


Figure 19: Schematic With Net Names Added

## Creating Buses

Using a similar procedure to adding net names, create buses for the two counter modules by adding bus name and size to the count and din wires.

1. Select **Add** → **Net Names** or click the Add Net Name icon from the Tools toolbar.



Figure 20: Add Net Name icon

Next, add the following four buses (name and size) to the schematic: count1(3:0), count2(3:0), din1(3:0) and din2(3:0).

2. To add buses:
  - a. Type the bus name and size in the text box; for example, din1(3:0).
  - Note:** Leave the default options Name the branch and Keep the name.
  - b. Place the cursor, which now displays the bus name and size, at the end of the hanging bus.
  - c. Click the left mouse button.
3. Press **Esc** to exit Add Net Name mode.

After adding the bus names to the counters, your schematic diagram should look like the following.

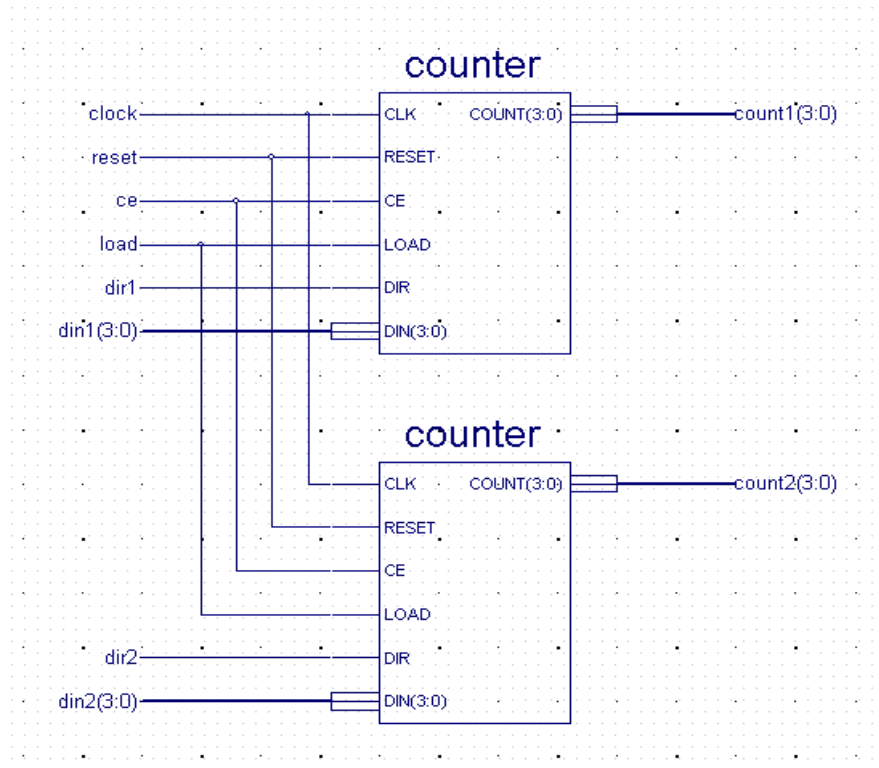


Figure 21: Schematic with Buses Added

## Adding I/O Markers

Next, identify the polarity of each signal (represented by a hanging wire) in these two counters. In this tutorial, you will add input and bidirectional signal markers to the schematic diagram. The results are shown in the schematic diagram in [Figure 23](#).

To add I/O markers:

1. Select **Add** → **I/O Marker** or click the Add I/O Marker icon from the Tools toolbar.



Figure 22: Add I/O Marker icon

2. Add input markers to the clock, reset, ce, load, dir1 and dir2 wires, and the din1(3:0) and din2(3:0) buses as follows:
  - a. Select the Add an input marker radio button on the Options tab.
  - b. Place the cursor, which now displays the input graphic, at the end of the counter input wire.
  - c. Click the left mouse button to add the marker.

The input graphic is added to the end of the wire, around the net or bus name.

**Note:** Click the cursor at the end of the hanging wire when adding the marker. If you try to add a marker to any location other than the end point, an error message box appears.

3. Add bidirectional markers to the count wires as follows:
  - a. Select the Add a bidirectional marker radio button on the Options tab.
  - b. Place the cursor, which now displays a bidirectional graphic, at the end of the counter outputs.
  - c. Click the left mouse button to add the marker.

Your completed schematic should look like the following diagram.

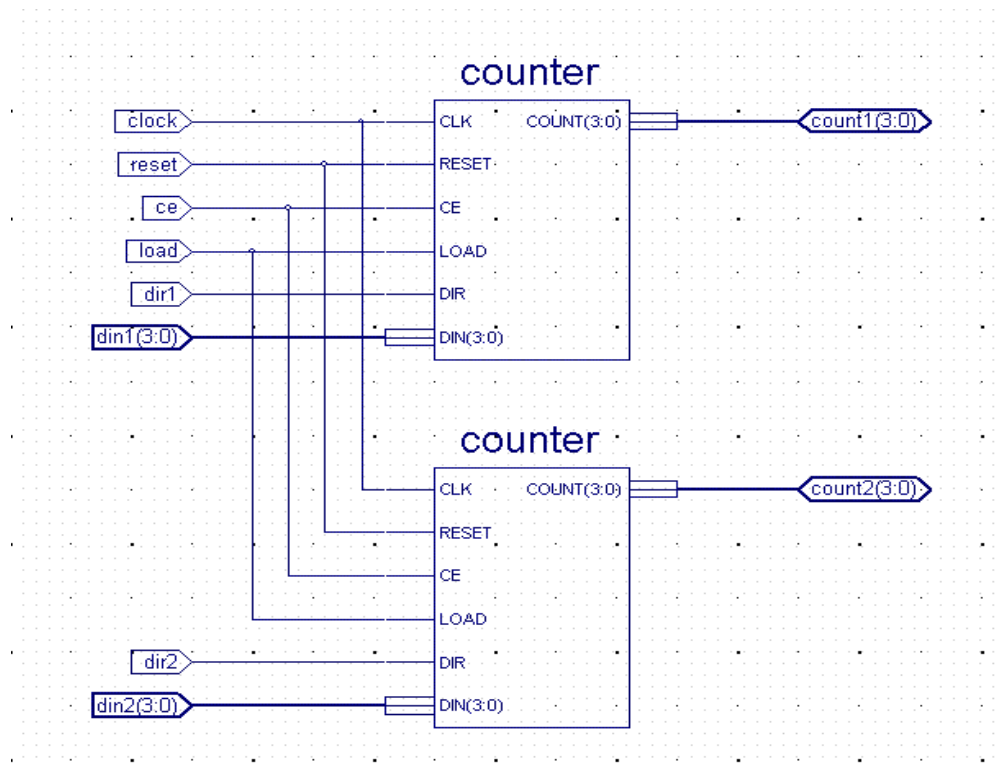


Figure 23: Completed Schematic

4. Save the schematic diagram using **File** → **Save**.
5. Exit ECS.

## Design Implementation

For this tutorial, design implementation covers two tasks: running the Implement Design process in Project Navigator, and viewing the resulting placed and routed design in Floorplanner.

**Note:** For more information about implementing a design, see *ISE Help*. Select **Help** → **ISE Help Contents**, expand either the FPGA or CPLD hierarchy in the left pane and expand the Implementing a Design hierarchy.

## Running Implement Design

First, run all processes (Synthesis through Place & Route) associated with the counter. To do so, run Implement Design on the schematic file:

1. Select top (top.sch) in the Sources in Project window.
2. Double-click **Implement Design** in the Processes for Source “top” window.

This runs all processes.

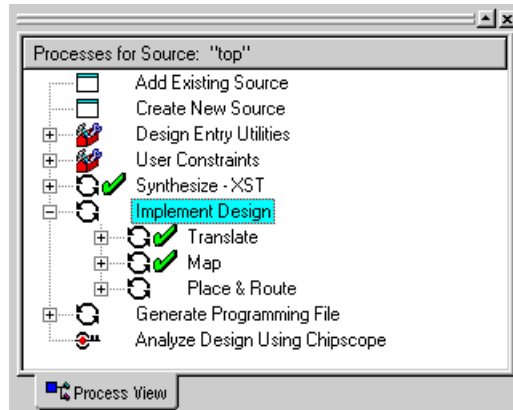


Figure 24: Implement Design processes

A check mark in the Processes for Source window denotes a process that was run successfully. An exclamation mark indicates that the process was run and that there is a warning for the process. More information about warnings can be obtained in the Transcript window.

## Viewing the Design in Floorplanner

Now, you can view the implemented design in Floorplanner. The Floorplanner is a graphical tool in which you can view and change the design hierarchy, floorplan, and perform design rule checks.

1. In Project Navigator, select top (top.sch) in the Sources in Project window.
2. In the Processes for Source window, click the + sign beside **Implement Design** and the + sign beside **Place & Route**.
3. Double-click **View/Edit Placed Design (Floorplanner)**.

The Floorplanner tool is launched and displays the placement of the design for the project.

To view the implemented design results in a more meaningful way, you can display and zoom in on the input/output signals.

1. In the top.fnf Design Hierarchy window (**View → Hierarchy**), select (highlight) the top-level hierarchy, ‘top (22 IOBs, 13 FGs, 8 CYS, 8 DFFs, 1 BUFG)’, to show the signals in the Placement window.

**Note:** Alternatively, you can draw a rectangle around the design area in the Placement window to show the signals.

2. Select **View → Zoom → In** or click the Zoom In icon.
3. 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.

**Note:** Alternatively, you can view isolated signals in the placement window by selecting individual signals from the list in the top.fnf Design Hierarchy window.

The placement in Floorplanner should look like the following.

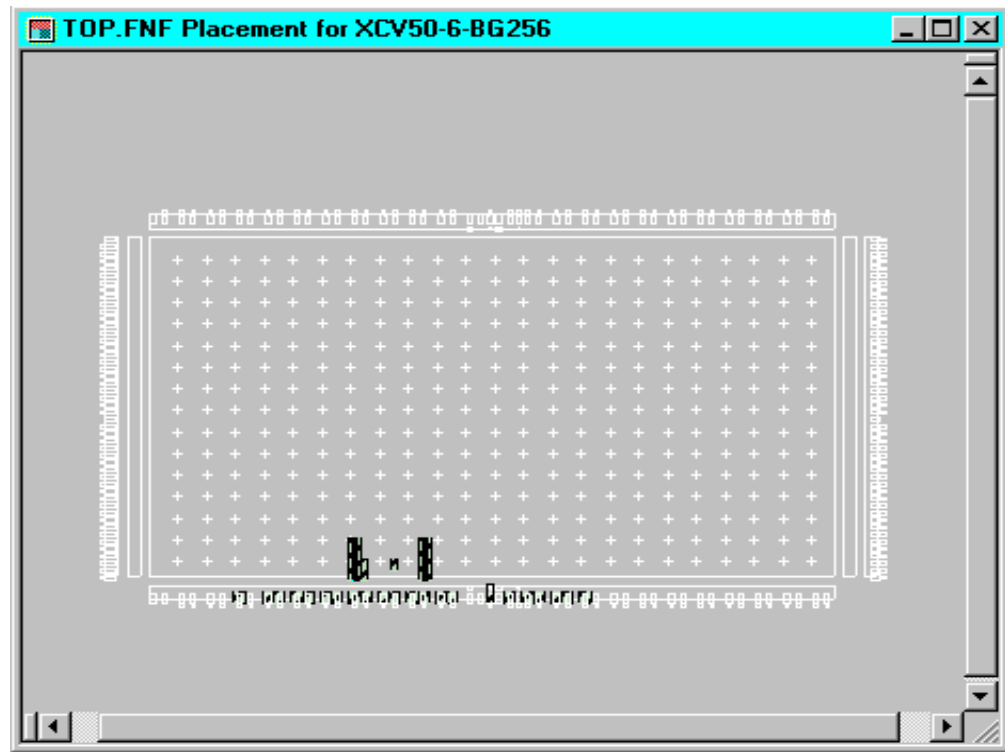


Figure 25: I/O Connections in Floorplanner

When you have finished viewing the implemented design:

1. Save the Floorplanner design view using **File** → **Save**.
2. Save top.fnf and then save the top.ucf file, created by ISE.

**Note:** top.ucf is a constraints file. It ensures that any constraints you have specified are saved in case of a subsequent implementation run.

3. Exit Floorplanner.

## Simulating the Top-level Design

Next, run a timing simulation on the top-level design created in the previous sections. Simulating the top-level design enables you to verify the logic and timing of the design. First, create a test bench waveform for the top-level design using HDL Bench, and then simulate the top-level design using ModelSim.

### Creating a Test Bench Waveform Source

Create a test bench waveform which you will modify in HDL Bench.

1. In Project Navigator, select top (top.sch) in the Sources in Project window.
2. Select **Project** → **New Source**.
3. In the New dialog box, select the **Test Bench Waveform** source type.

4. Type the name 'top\_tbw'.
5. Select **Next**.
6. Ensure that top is the associated source and select **Next**.
7. Select **Finish**.  
HDL Benchner is launched.
8. Click **OK** to use the default timing constraints for the test bench waveform.

## Initializing Counter Inputs

**Note:** Before initializing the inputs, ensure that you have Radix set to Decimal. Radix is set to Decimal when the 10 icon in the HDL Benchner toolbar is selected.

In the waveform in HDL Benchner, initialize the counter inputs as follows. Verify your entries using Figure 26.

**Note:** Enter the input stimulus in the blue area in each cell.

1. Click the ce cell under clock cycle 3 until it is set high.
2. Click the dir1 cell under clock cycle 2 until it is set high.
3. Click the dir2 cell under clock cycle 1 until it is set high.
4. Click the dir2 cell under clock cycle 2 until it is set low.
5. Click the reset cell under clock cycle 1 until it is set high.
6. Click the reset cell under clock cycle 2 until it is set low.

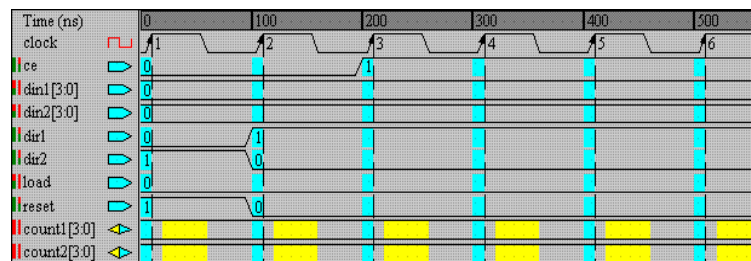


Figure 26: HDL Benchner Stimulus and Response Entries

## Generating the Expected Responses

To generate the expected response, make the following response entries in the yellow areas in the waveform in HDL Benchner.

1. Click the yellow count1(3:0) cell under clock cycle 2.
2. Click the Pattern button to launch the Pattern Wizard.

3. Set the parameters in the Pattern Wizard dialog box so that the expected output counts from 0 to 7 as follows:

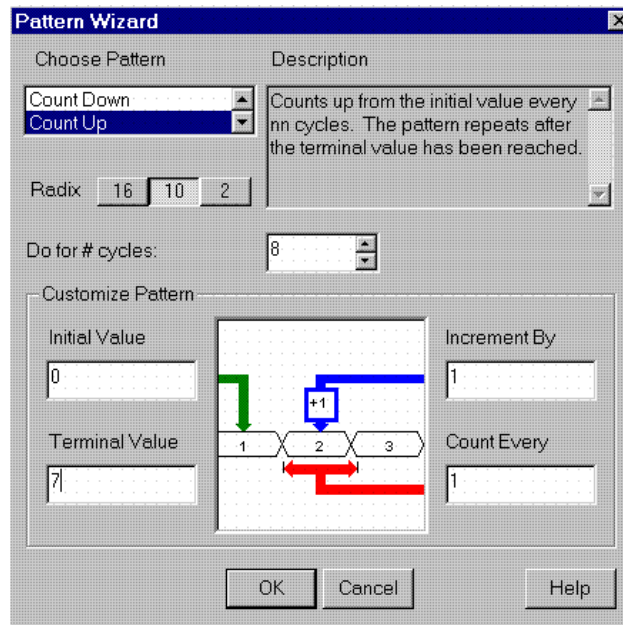


Figure 27: Pattern Wizard Settings

4. Click **OK** in the Pattern Wizard dialog box.
5. Click the yellow count2(3:0) cell under clock cycle 2 and enter 0 (zero).
6. Click the yellow count2(3:0) cell under clock cycle 3.
7. Click the Pattern button to launch the Pattern Wizard.



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

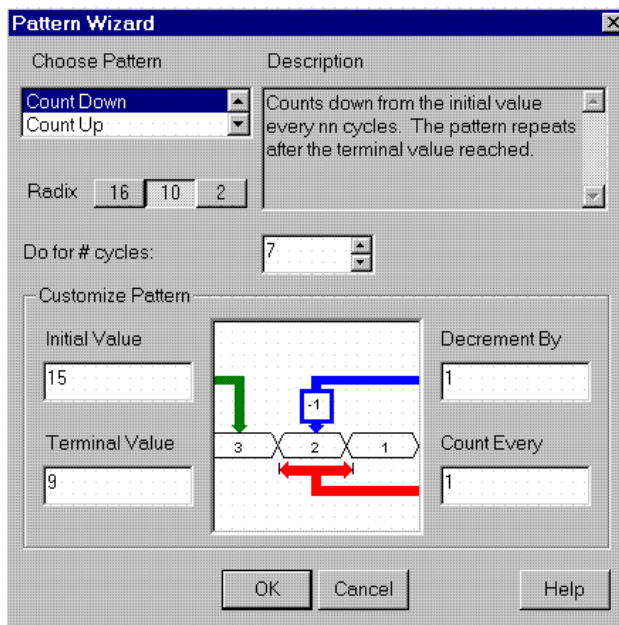


Figure 28: Pattern Wizard Settings

9. Click **OK** in the Pattern Wizard dialog box.

The test bench waveform should look like the following.

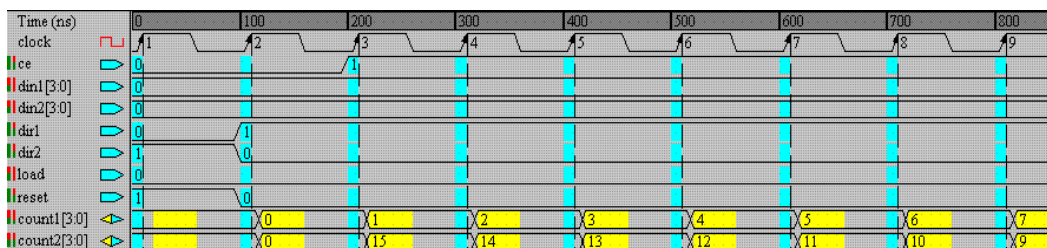


Figure 29: Test Bench Waveform

10. Save your test bench waveform by clicking the Save Waveform icon.



Figure 30: Save Waveform icon

- 11.
12. Exit HDL Bench.

In the Source for Project window in Project Navigator, the new test bench waveform file is a subset of top (top.sch).

## Post-place and Route Simulation

To perform post-place and route simulation on the top-level design:

1. In Project Navigator, select top\_tbw.tbw in the Sources in Project window.
2. In the Processes for Source "top.tbw" window, double-click **Simulate Post-Place and Route VHDL Model** found in the **ModelSim Simulator** hierarchy.  
ModelSim is launched with the back-annotated design.
3. Click **Zoom** → **Zoom Full** or click the Zoom Full icon in the toolbar.



Figure 31: Zoom Full icon

The waveform should look like the following.

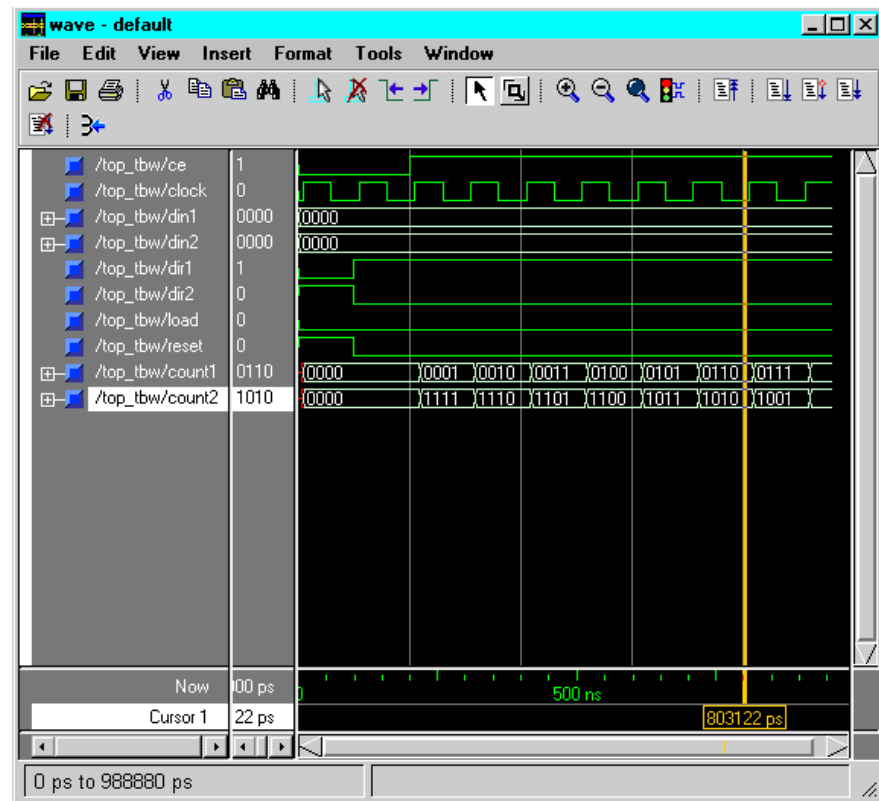


Figure 32: Timing Simulation Waveform

4. Verify that the time simulation passes a 10ns clock time delay.
5. When you have finished analyzing your results, exit ModelSim by closing the main ModelSim window.

# *EDIF Design Flow*

---

This appendix explains how to implement a design in ISE from an EDIF source file.

This appendix contains the following sections.

- [“Design Entry”](#)
- [“Design Implementation”](#)

## EDIF Overview

Once you have completed the EDIF Design Flow you will know how to do the following:

- Import your own netlist (EDIF file) in [“EDIF Design Flow”](#)
- View the placed and routed design in FPGA Editor in [“EDIF Design Flow”](#)

## Design Entry

To add an EDIF source file to a new project in Project Navigator, first create a new project, add the EDIF source file, then implement the design.

### Creating a New Project

To create a new project using an EDIF source file:

1. Select **File** → **New Project**.
2. In the New Project Wizard Project dialog box, type the desired location in the Project Location field, or browse to the directory under which you want to create your new project directory using the browse button next to the Project Location field.
3. Enter ‘TutorialEDIF’ in the Project Name field.

When you enter ‘TutorialEDIF’ in the Project Name field, a TutorialEDIF directory is automatically created in the directory path in the Project Location field. For example, for the directory path C:\My\_Projects, entering the Project Name ‘TutorialEDIF’ modifies the path to be C:\My\_Projects\TutorialEDIF.

4. Use the pull-down arrow to select EDIF from the Top-Level Module Type field. Click in the field to access the pull-down list.

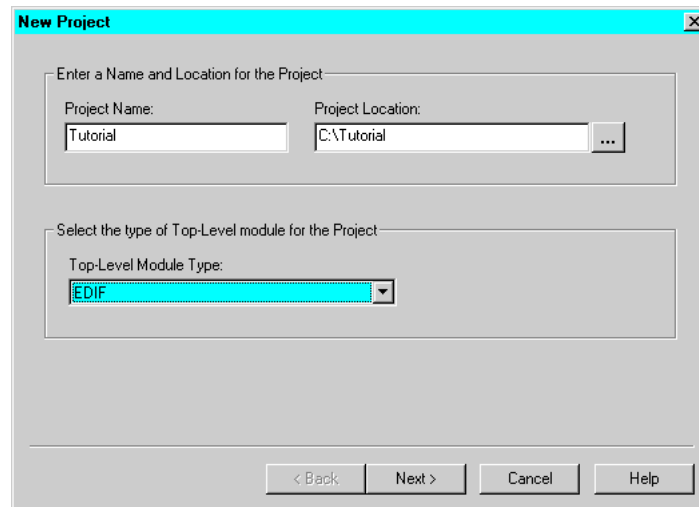


Figure A-1: New Project Wizard Project Dialog Box

5. Click **Next**  
Now, you will add an EDIF source file to the new project using the New Project Wizard. The EDIF source file used in this tutorial is an example file that is shipped with the ISE software. To add the example file to the project:
6. Type the full path in the Input Design field, or browse to the edif\_flow directory found in the Xilinx install directory:  
%XILINX%/ISEexamples/edif\_flow
7. Select the file 'mf.edn'.
8. Click **Open**.
9. Select the Copy Input Design to the Project Directory box (default).

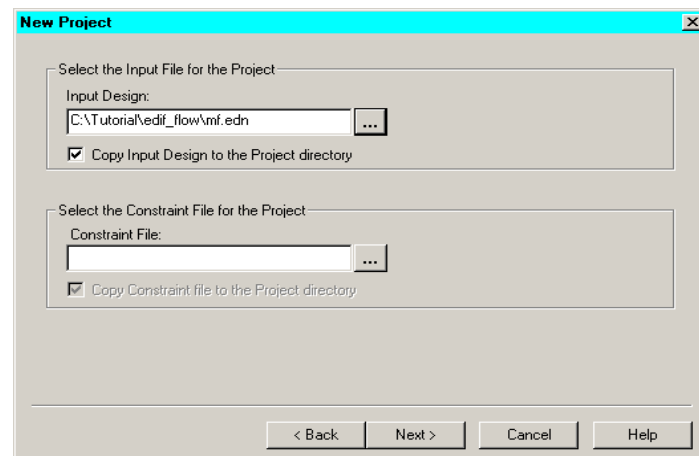


Figure A-2: New Project Wizard Dialog Box

10. Click **Next**

11. In the New Project Wizard Device and Design Flow dialog box, use the pull-down arrow to select the Value for each Property Name. Click in the field to access the pull-down list.

Change the values as follows:

- ◆ Device Family: Virtex
- ◆ Device: xcv300
- ◆ Package: bg352
- ◆ Speed Grade: -6
- ◆ Simulator: Modelsim.

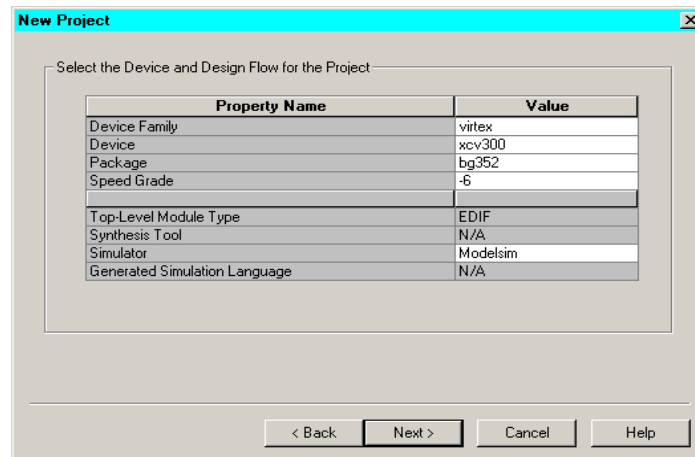


Figure A-3: New Project Wizard Device and Design Flow Dialog Box

12. Click **Next**
13. Click **Finish**

A copy of the file is now added to the newly created project.

## Design Implementation

Next, implement the design and use FPGA Editor to view the placed and routed design.

### Running Implement Design

To run design implementation on the design:

1. Select the EDIF file 'mf.edn' from the Sources in Project window.
2. Double-click **Implement Design** in the Processes for Source window.

This runs all processes (Translate through Place & Route) required to implement the design. You will subsequently open the implemented design in FPGA Editor.

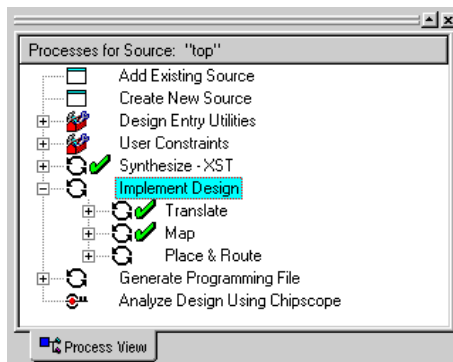


Figure A-4: Implement Design processes

A check mark in the Processes for Source “mf.edn” window denotes a process that was run successfully. An exclamation mark indicates that the process was run and that there is a warning for the process. More information about warnings can be obtained in the Transcript window.

## Viewing the Design in FPGA Editor

FPGA Editor is a graphical application used to display and configure FPGAs. To view the design from ‘mf.edn’ in FPGA Editor:

1. When implementation is finished, in the Processes for Source “mf.edn” window, click the + sign beside **Implement Design** and click the + sign beside **Place & Route**.
2. Double-click **View/Edit Routed Design (FPGA Editor)**.

The placed design, mf.ncd, is automatically displayed in FPGA Editor.

3. Click on the Type column header in the List1 window to list all components by type. Resize the table and the columns as desired to read the column names clearly. Verify that all 26 I/Os are accounted for as IOBs in the Type column.

**Note:** Clock I/Os will be listed under the GCLK type, not the IOB type.

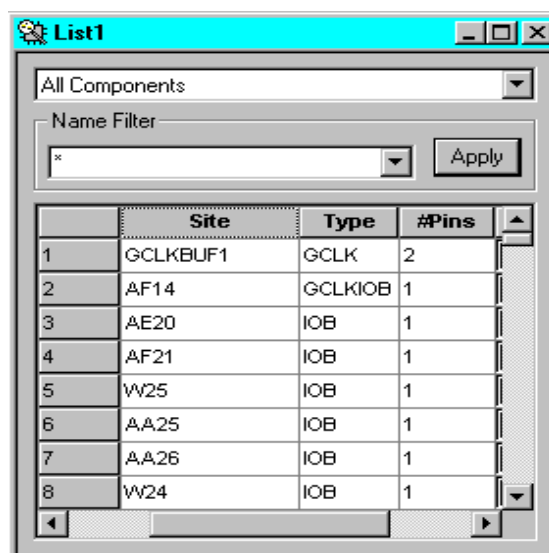


Figure A-5: Listing all components by type

When finished viewing the design, exit FPGA Editor.





# Index

---

## B

behavioral simulation  
    HDL Bencher 19, 31  
    ModelSim 20  
behavioral statements 15  
buses in schematics 26

## C

comment lines 16  
constraints  
    timing 18, 31  
Create Schematic Symbol 22

## D

design entry  
    EDIF netlist 35  
    schematics 22  
    VHDL counter module 13  
design flow  
    XST VHDL 14  
design implementation  
    example 28, 37

## E

ECS schematic window 23

## F

F1 help 12  
Floorplanner  
    viewing design 29  
FPGA Editor  
    viewing design 38

## G

Generate Expected Simulation Results 19

## H

HDL Bencher  
    creating waveform 18, 31  
HDL Editor 14  
    editing module 15  
help  
    online help 12

## I

I/O markers  
    adding 27  
Implement Design 29, 37  
instantiation  
    VHDL modules 23  
ISE help 12  
ISE software  
    starting 12  
    stopping 12

## L

Language Templates 15

## M

markers  
    adding 27  
ModelSim simulation 20, 33

## N

net names  
    adding to wires 25  
new project  
    EDIF design flow 35  
    VHDL design flow 13  
New Project dialog box 13

## O

online help 12

## P

port definitions 15  
project  
    creating new 13, 35  
Project Location field 13, 35  
Project Name field 13, 35  
Property Name field 14, 36

## S

schematic symbols 22  
schematics  
    adding I/O markers 27  
    creating buses 26  
    instantiating VHDL modules 23  
    wiring 24  
Simulate Behavioral VHDL Model 20  
Simulate Post-Place & Route VHDL Module 21, 33  
simulation  
    behavioral 19, 20, 31  
    timing 21, 33  
starting ISE software 12  
stopping ISE software 12

## T

test bench 17  
timing constraints 18  
timing simulation 21, 33  
top-level schematics 22

## V

VHDL module  
    behavioral statements 15  
    complete 17  
    creating 14  
    creating schematic symbols 22  
    editing contents 15  
    instantiating in schematics 23  
    port definitions 15

View/Edit Placed Design (Floor-planner) 29  
View/Edit Routed Design (FPGA Editor) 38  
Virtex  
    device family entry 14

## W

waveform  
    creating 18, 31  
    saving 19  
wires  
    in schematics 24

## X

XST VHDL  
    design flow entry 14