

# ***ISE 4 Tutorial***



The Xilinx logo shown above is a registered trademark of Xilinx, Inc.

CoolRunner, RocketChips, RocketIP, Spartan, StateBENCH, StateCAD, Virtex, XACT, XILINX, 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, Bencher, 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 Bencher, IRL, J Drive, JBits, LCA, LogiBLOX, Logic Cell, LogiCORE, LogicProfessor, MicroBlaze, MicroVia, MultiLINX, NanoBlaze, PicoBlaze, PLUSASM, PowerGuide, PowerMaze, QPro, Real-PCI, Rocket I/O, Select I/O, SelectRAM, SelectRAM+, Silicon Xpresso, Smartguide, Smart-IP, SmartSearch, SMARTswitch, System ACE, Testbench In A Minute, TrueMap, UIM, VectorMaze, VersaBlock, VersaRing, Wave Table, WebFITTER, WebPACK, WebPOWERED, XABEL, XACT<sup>step</sup> Advanced, XACT<sup>step</sup> Foundry, XACT-Floorplanner, XACT-Performance, XAM, XAPP, X-BLOX +, XChecker, XDM, XEPLD, Xilinx Foundation Series, Xilinx XDTV, Xinfo, XSI, XtremeDSP, all XC designated products, 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, Inc. devices and products are protected under one or more of the following U.S. Patents: 4,642,487; 4,695,740; 4,706,216; 4,713,557; 4,746,822; 4,750,155; 4,758,985; 4,820,937; 4,821,233; 4,835,418; 4,855,619; 4,855,669; 4,902,910; 4,940,909; 4,967,107; 5,012,135; 5,023,606; 5,028,821; 5,047,710; 5,068,603; 5,140,193; 5,148,390; 5,155,432; 5,166,858; 5,224,056; 5,243,238; 5,245,277; 5,267,187; 5,291,079; 5,295,090; 5,302,866; 5,319,252; 5,319,254; 5,321,704; 5,329,174; 5,329,181; 5,331,220; 5,331,226; 5,332,929; 5,337,255; 5,343,406; 5,349,248; 5,349,249; 5,349,250; 5,349,691; 5,355,035; 5,357,153; 5,360,747; 5,361,229; 5,362,999; 5,365,125; 5,367,207; 5,386,154; 5,394,104; 5,397,943; 5,399,924; 5,399,925; 5,406,133; 5,410,189; 5,410,194; 5,414,377; 5,422,833; 5,426,378; 5,426,379; 5,430,687; 5,432,719; 5,448,181; 5,448,493; 5,450,021; 5,450,022; 5,453,706; 5,455,525; 5,466,117; 5,469,003; 5,475,253; 5,477,414; 5,481,206; 5,483,478; 5,486,707; 5,486,776; 5,488,316; 5,489,858; 5,489,866; 5,491,353; 5,495,196; 5,497,108; 5,498,979; 5,498,989; 5,499,192; 5,500,608; 5,500,609; 5,502,000; 5,502,440; 5,504,439; 5,504,440; 5,506,518; 5,506,523; 5,506,878; 5,513,124; 5,517,135; 5,521,835; 5,521,837; 5,523,963; 5,523,971; 5,524,097; 5,526,322; 5,528,169; 5,528,176; 5,530,378; 5,530,384; 5,546,018; 5,550,839; 5,550,843; 5,552,722; 5,553,001; 5,559,751; 5,561,367; 5,561,629; 5,561,631; 5,563,527; 5,563,528; 5,563,529; 5,563,827; 5,565,792; 5,566,123; 5,570,051; 5,570,059; 5,574,634; 5,574,655; 5,578,946; 5,581,198; 5,581,199; 5,581,738; 5,583,450; 5,583,452; 5,592,105; 5,594,367; 5,598,424; 5,600,263; 5,600,264; 5,600,271; 5,600,597; 5,608,342; 5,610,536; 5,610,790; 5,610,829; 5,612,633; 5,614,844; 5,617,021; 5,617,041; 5,617,327; 5,617,573; 5,623,387; 5,627,480; 5,629,637; 5,629,886; 5,631,577; 5,631,583; 5,635,851; 5,636,368; 5,640,106; 5,642,058; 5,646,545; 5,646,547; 5,646,564; 5,646,903; 5,648,732; 5,648,913; 5,650,672; 5,650,946; 5,652,904; 5,654,631; 5,654,665; 5,656,950; 5,657,290; 5,659,484; 5,661,660; 5,661,685; 5,668,495; 5,670,896; 5,670,897; 5,672,966; 5,673,198; 5,675,262; 5,675,270; 5,675,589; 5,677,638; 5,682,107; 5,684,413; 5,689,133; 5,689,516; 5,691,907; 5,691,912; 5,694,047; 5,694,055; 5,694,056; 5,694,399; 5,696,454; 5,701,091; 5,701,441; 5,703,759; 5,705,932; 5,705,938; 5,708,597; 5,712,579; 5,714,890; 5,715,197; 5,717,340; 5,719,506; 5,719,507; 5,724,276; 5,726,484; 5,726,584; 5,734,866; 5,734,868; 5,737,234; 5,737,235; 5,737,631; 5,742,178; 5,742,179; 5,742,531; 5,744,974; 5,744,979; 5,744,981; 5,744,995; 5,748,942;

---

5,748,979; 5,752,006; 5,752,035; 5,754,459; 5,758,192; 5,760,603; 5,760,604; 5,760,607; 5,761,483; 5,764,076; 5,764,534; 5,764,564; 5,768,179; 5,770,951; 5,773,993; 5,778,439; 5,781,756; 5,784,313; 5,784,577; 5,786,240; 5,787,007; 5,789,938; 5,790,479; 5,790,882; 5,795,068; 5,796,269; 5,798,656; 5,801,546; 5,801,547; 5,801,548; 5,808,479; 5,811,985; 5,815,004; 5,815,016; 5,815,404; 5,815,405; 5,818,255; 5,818,730; 5,821,772; 5,821,774; 5,825,202; 5,825,662; 5,825,787; 5,828,230; 5,828,231; 5,828,236; 5,828,608; 5,831,448; 5,831,460; 5,831,845; 5,831,907; 5,835,402; 5,838,167; 5,838,901; 5,838,954; 5,841,296; 5,841,867; 5,844,422; 5,844,424; 5,844,829; 5,844,844; 5,847,577; 5,847,579; 5,847,580; 5,847,993; 5,852,323; 5,861,761; 5,862,082; 5,867,396; 5,870,309; 5,870,327; 5,870,586; 5,874,834; 5,875,111; 5,877,632; 5,877,979; 5,880,492; 5,880,598; 5,880,620; 5,883,525; 5,883,852; 5,886,538; 5,889,411; 5,889,412; 5,889,413; 5,889,701; 5,892,681; 5,892,961; 5,894,420; 5,896,047; 5,896,329; 5,898,319; 5,898,320; 5,898,602; 5,898,618; 5,898,893; 5,907,245; 5,907,248; 5,909,125; 5,909,453; 5,910,732; 5,912,937; 5,914,514; 5,914,616; 5,920,201; 5,920,202; 5,920,223; 5,923,185; 5,923,602; 5,923,614; 5,928,338; 5,931,962; 5,933,023; 5,933,025; 5,933,369; 5,936,415; 5,936,424; 5,939,930; 5,940,606; 5,942,913; 5,944,813; 5,945,837; 5,946,478; 5,949,690; 5,949,712; 5,949,983; 5,949,987; 5,952,839; 5,952,846; 5,955,888; 5,956,748; 5,958,026; 5,959,821; 5,959,881; 5,959,885; 5,961,576; 5,962,881; 5,963,048; 5,963,050; 5,969,539; 5,969,543; 5,970,142; 5,970,372; 5,971,595; 5,973,506; 5,978,260; 5,986,958; 5,990,704; 5,991,523; 5,991,788; 5,991,880; 5,991,908; 5,995,419; 5,995,744; 5,995,988; 5,999,014; 5,999,025; 6,002,268; 6,002,282; 6,002,991; 6,005,423; 6,005,829; 6,008,666; 6,011,407; 6,011,740; 6,016,063; 6,018,250; 6,018,624; 6,020,633; 6,020,756; 6,020,757; 6,020,776; 6,021,423; 6,023,564; 6,023,565; 6,025,736; 6,026,481; 6,028,445; 6,028,450; 6,033,938; 6,034,542; 6,034,548; 6,034,557; 6,035,106; 6,037,800; 6,038,386; 6,041,340; 6,043,692; 6,044,012; 6,044,025; 6,046,603; 6,047,115; 6,049,222; 6,049,227; 6,051,992; 6,054,871; 6,055,205; 6,057,589; 6,057,704; 6,057,708; 6,061,417; 6,061,418; 6,067,508; 6,069,488; 6,069,489; 6,069,490; 6,069,849; 6,070,260; 6,071,314; 6,072,348; 6,073,154; 6,074,432; 6,075,418; 6,078,201; 6,078,209; 6,078,528; 6,078,735; 6,078,736; 6,081,914; 6,084,429; 6,086,629; 6,086,631; 6,091,262; 6,091,263; 6,091,892; 6,094,063; 6,094,065; 6,094,385; 6,097,210; 6,097,238; 6,099,583; 6,100,705; 6,101,132; 6,101,143; 6,104,211; 6,105,105; 6,107,821; 6,107,826; 6,107,827; 6,112,322; 6,114,843; 6,118,286; 6,118,298; 6,118,300; 6,118,324; 6,118,869; 6,118,938; 6,120,549; 6,120,551; 6,121,795; 6,124,724; 6,124,731; 6,130,550; 6,133,751; 6,134,191; 6,134,517; 6,137,307; 6,137,714; 6,144,220; 6,144,225; 6,144,262; 6,144,933; 6,150,838; 6,150,839; 6,150,863; 6,154,048; 6,154,049; 6,154,052; 6,154,053; 6,157,209; 6,157,211; 6,157,213; 6,160,418; 6,160,431; 6,163,167; 6,167,001; 6,167,416; 6,167,545; 6,167,558; 6,167,560; 6,172,518; 6,172,519; 6,172,520; 6,173,241; 6,175,246; 6,175,530; 6,177,819; 6,177,830; 6,181,158; 6,181,164; 6,184,708; 6,184,709; 6,184,712; 6,185,724; 6,188,091; 6,191,610; 6,191,613; 6,191,614; 6,192,436; 6,195,774; 6,199,192; 6,201,406; 6,201,410; 6,201,411; and 6,202,106; Re. 34,363, Re. 34,444, and Re. 34,808. 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.

Copyright 1991-2001 Xilinx, Inc. All Rights Reserved.



# About This Manual

---

## Manual Contents

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

## 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 <a href="http://support.xilinx.com/support/techsup/tutorials/index.htm">http://support.xilinx.com/support/techsup/tutorials/index.htm</a>
Answers Database	Current listing of solution records for the Xilinx software tools Search this database using the search function at <a href="http://support.xilinx.com/support/searchtd.htm">http://support.xilinx.com/support/searchtd.htm</a>
Application Notes	Descriptions of device-specific design techniques and approaches <a href="http://support.xilinx.com/apps/appswb.htm">http://support.xilinx.com/apps/appswb.htm</a>
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>
Xcell Journals	Quarterly journals for Xilinx programmable logic users <a href="http://support.xilinx.com/xcell/xcell.htm">http://support.xilinx.com/xcell/xcell.htm</a>
Technical Tips	Latest news, design tips, and patch information for the Xilinx design environment <a href="http://support.xilinx.com/support/techsup/journals/index.htm">http://support.xilinx.com/support/techsup/journals/index.htm</a>

# Conventions

---

This manual uses the following conventions of style. An example illustrates most conventions.

## 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 loc2 ... locn;
```

## Online Document

The following conventions are used for online documents.

- **Blue text** indicates cross-references within a book. Red text indicates cross-references to other books. Click the colored text to jump to the specified cross-reference.
- **Blue, underlined text** indicates a Web site. Click the link to open the specified Web site. You must have a Web browser and internet connection to use this feature.

# Contents

---

## About This Manual

Manual Contents .....	v
Additional Resources .....	vi

## Conventions

Typographical .....	vii
Online Document .....	viii

## ISE Tutorial

Tutorial Overview .....	1-2
Getting Started .....	1-3
Software Requirements .....	1-3
Starting the ISE Software .....	1-3
Accessing Online Help .....	1-3
Design Entry (VHDL) .....	1-4
Creating a New Project .....	1-4
Creating a Counter Module .....	1-5
Modifying Counter Module with Counter Template .....	1-6
Simulating the Behavioral Model .....	1-9
Creating a Testbench Waveform Source .....	1-9
Initializing Counter Inputs .....	1-10
Generating the Expected Simulation Output Values .....	1-11
Simulating with ModelSim .....	1-12
Behavioral Simulation .....	1-12
Post-place and Route Simulation .....	1-14
Design Entry (Top-Level Schematic) .....	1-16
Creating a Schematic Symbol for the VHDL Module .....	1-16
Creating a New Top-Level Schematic .....	1-16
Instantiating VHDL Modules .....	1-16
Wiring the Schematic .....	1-18
Adding Net Names to Wires .....	1-19
Creating Buses .....	1-21
Adding I/O Markers .....	1-22
Design Implementation .....	1-25
Running Implement Design .....	1-25
Viewing the Design in Floorplanner .....	1-26

Simulating the Top-level Design .....	1-28
Creating a Testbench Waveform Source .....	1-28
Initializing Counter Inputs .....	1-28
Generating the Expected Responses .....	1-29
Post-place and Route Simulation .....	1-31

## **Appendix: EDIF Design**

Design Entry .....	A-1
Creating a New Project .....	A-1
Adding the EDIF source file .....	A-2
Design Implementation .....	A-3
Running Implement Design .....	A-3
Viewing the Design in FPGA Editor .....	A-4

## **Index**

# ISE Tutorial

---

The ISE 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.

**Note** This tutorial is designed for ISE 4.x, PC version.

This tutorial contains the following sections.

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

To learn how to import your own netlist into ISE and view the design, see “[Appendix: EDIF Design.](#)”

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

## 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 4.x
- ModelSim VHDL

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 4.x** → **Project Navigator**.

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

### Accessing Online 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. Online help is available by pressing F1. When you press F1, the help system for the tool in which you are working is launched. For example, when you press F1 while in ECS, ECS help is displayed.

## Design Entry (VHDL)

In this section, you will create a 4-bit counter module using the Language Templates. To do so, first create a new project and counter module, then modify the counter module with the counter template.

### Creating a New Project

To create a new project:

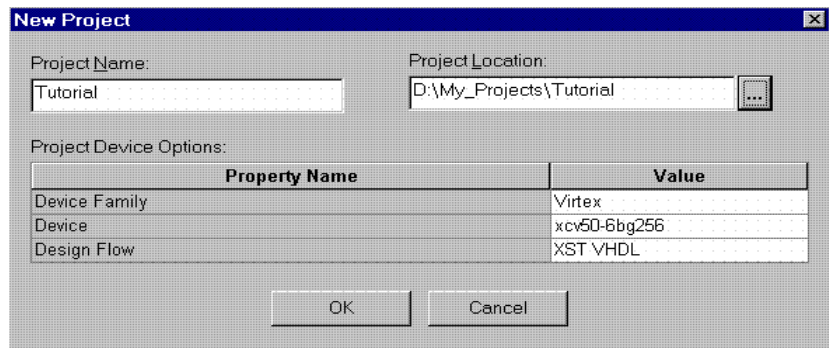
1. Select **File** → **New Project**.
2. In the New Project dialog box, type the desired location in the Project Location field, or browse to the directory under which you want to create your new project directory using the browse button next to the Project Location field.
3. Enter 'Tutorial' in the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

4. 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: xcv50-6bg256
- ◆ Design Flow: XST VHDL



**Figure 1-1 New Project Dialog Box**

5. Click **OK**.

ISE creates and displays the new project in Project Navigator.

## Creating a Counter Module

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

1. Select **Project** → **New Source**.
2. Select VHDL Module as the source type.
3. Type in the file name 'counter'.
4. Click **Next**.
5. Click **Next**.
6. Click **Finish** to complete the new source file template.

Counter.vhd, which is displayed in the HDL Editor window, contains the library declaration and use statements along with the empty entity and architecture pair for the counter you have just created.

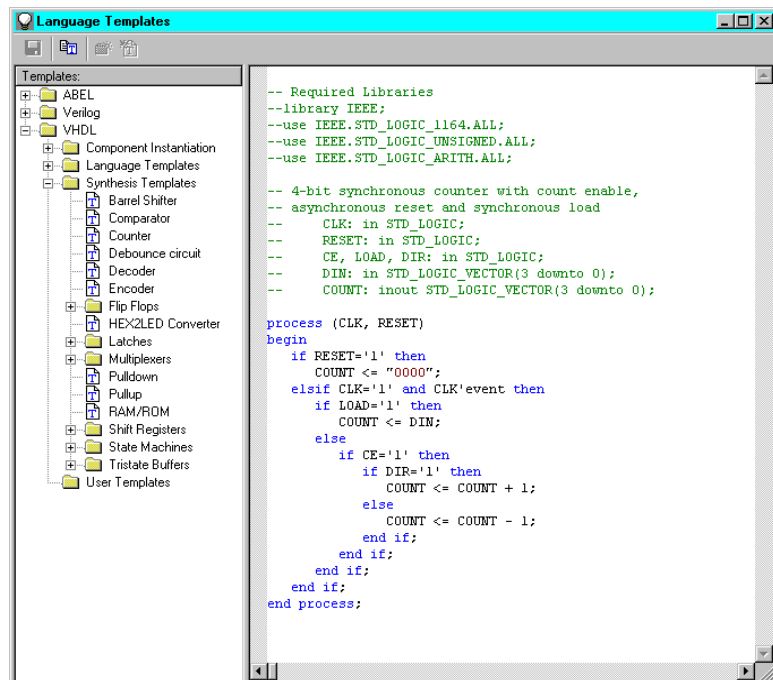
## Modifying Counter Module with Counter Template

To complete the counter module, insert port declarations and the behavioral code for the counter from the ISE Language Template.

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



2. In the Language Templates window, click the + sign next to VHDL to expand the hierarchy and then click the + sign next to Synthesis Templates.



**Figure 1-2 Counter Language Template**

3. Click and drag the Counter template from the VHDL Synthesis Templates folder, a subset of the VHDL folder, and drop it into counter.vhd between the begin and end behavioral statements.

4. Close the Language Templates window.
5. Cut the port definitions from the comment section of the counter.vhd file and paste them into the parentheses in the port declaration of the counter entity. The port definitions are the following lines:

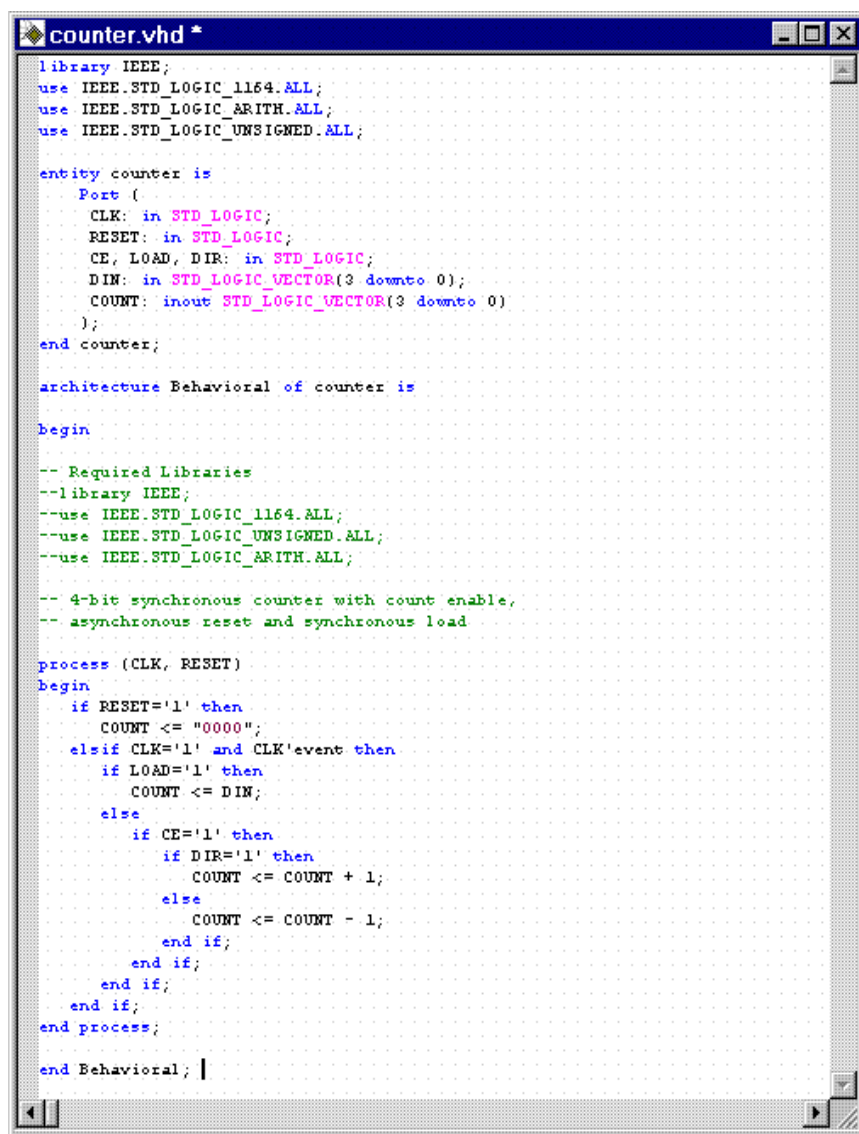
```
-- CLK: in STD_LOGIC;  
-- RESET: in STD_LOGIC;  
-- CE, LOAD, DIR: in STD_LOGIC;  
-- DIN: in STD_LOGIC_VECTOR(3 downto 0);  
-- COUNT: inout STD_LOGIC_VECTOR(3 downto 0);
```

6. Uncomment the above port definitions in your counter.vhd file by removing the dashes from the beginning of the line.
7. Remove the semicolon that follows the COUNT port definition.

```
COUNT: inout STD_LOGIC_VECTOR(3 downto 0)
```

8. Save counter.vhd by selecting **File** → **Save**.

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, LOAD, DIR: in STD_LOGIC;
        DIN: in STD_LOGIC_VECTOR(3 downto 0);
        COUNT: inout STD_LOGIC_VECTOR(3 downto 0)
    );
end counter;

architecture Behavioral of counter is

begin

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

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

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

end Behavioral;
```

Figure 1-3 Modified Counter Module

## Simulating the Behavioral Model

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

### Creating a Testbench Waveform Source

First, create a testbench waveform in Project Navigator which you will modify in HDL Benchner.

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

**Note** In other projects, you can associate your testbench waveform with other sources.

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

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

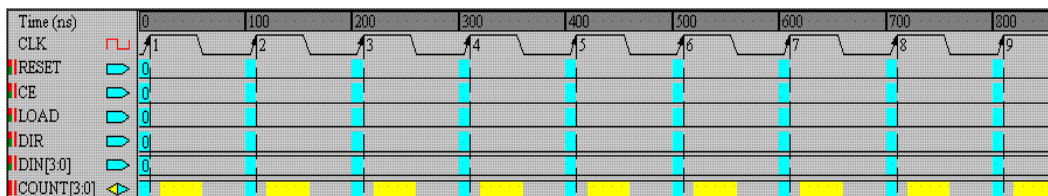
You will now specify the timing parameters used during simulation. The clock high time and clock low time together define the clock period for which the design must operate. The Input setup time defines when inputs must be valid. The Output valid delay defines the time after active clock edge when the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

8. Click **OK** to accept the default timing constraints.

Your testbench waveform should look like the following.



**Figure 1-4 Testbench waveform in HDL Benchner**

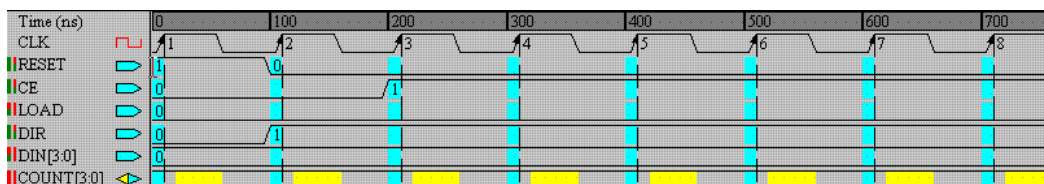
## Initializing Counter Inputs

In the waveform in HDL Benchner, 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 until the cell is set high.
2. Click the RESET cell under CLK cycle 2 until the cell is reset low.
3. Click the CE cell under CLK cycle 3 until it is set high.
4. Click the DIR cell under CLK cycle 2 until the cell is set high.

Your testbench waveform should now look like the following.



**Figure 1-5 HDL Benchner Stimulus and Response Entries**

5. Save your testbench waveform by selecting **File** → **Save Waveform** or by clicking the Save Waveform icon in the toolbar.



Next, HDL Benchner will prompt you to set the number of clock cycles for which you wish to simulate.

6. Enter 8 in the dialog box: 'End the testbench \_\_ cycles after the last input assignment'. Default value is 1.

This extends the waveform 8 clock cycles past the assertion of CE high.

7. Click **OK**.
8. Exit HDL Benchner.

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

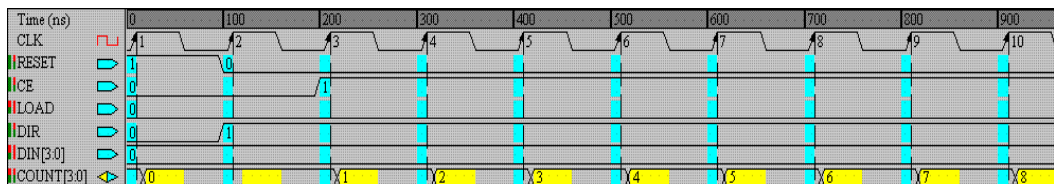
## Generating the Expected Simulation Output Values

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 Current 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 testbench waveform.

Your testbench waveform should look like the following.



**Figure 1-6 Generated Simulation Results**

4. Exit HDL Benchner without saving your waveform.

## Simulating with ModelSim

With the expected results generated in HDL Bench, you are now ready to run your simulation with ModelSim. 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.

1. In Project Navigator, select counter\_tbw.tbw in the Sources in Project window.
2. In the Processes for Current Source window, double-click Simulate Behavioral VHDL Model found in the ModelSim Simulator hierarchy.  
ModelSim is launched.
3. 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 until you reinstall or reconfigure ModelSim.

Your simulation results are displayed in the ModelSim wave window.

**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 testbench 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 time specified by the Simulation Run Time property (default is 1000 ns)
4. Click **Zoom** → **Zoom Full** or click the Zoom Full icon in the toolbar.

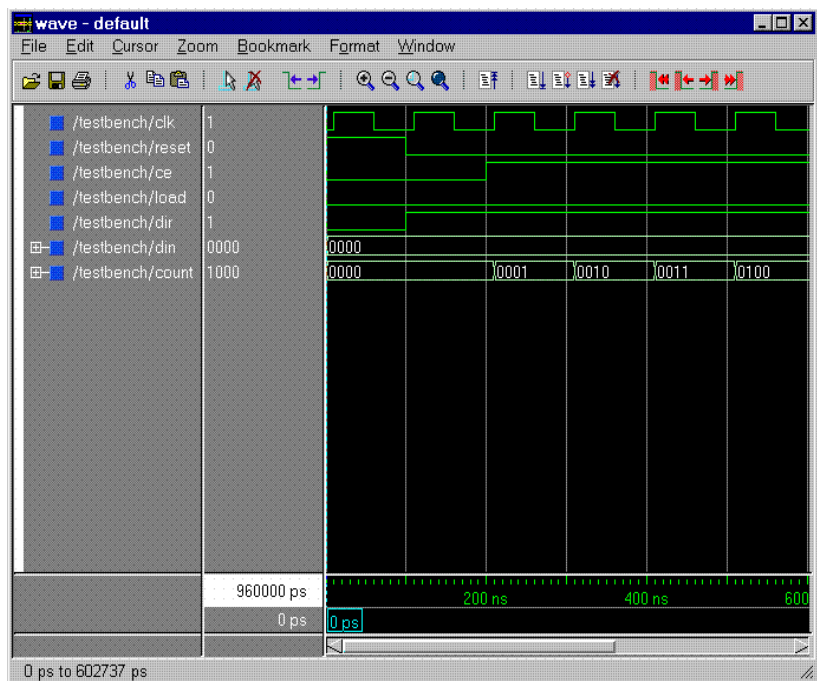


5. Click **Zoom** → **Zoom In** or click the Zoom In icon in the toolbar.



6. Scroll to the far left of the waveform.

The output waveform should look like that in [Figure 1-7](#).



**Figure 1-7 Behavioral Simulation Waveform**

7. 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 Current Source window, double-click Simulate Post-Place & Route VHDL Model found in the ModelSim Simulator hierarchy.

**Note** This will take the design through place-and-route and back-annotation.

ModelSim is launched.

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

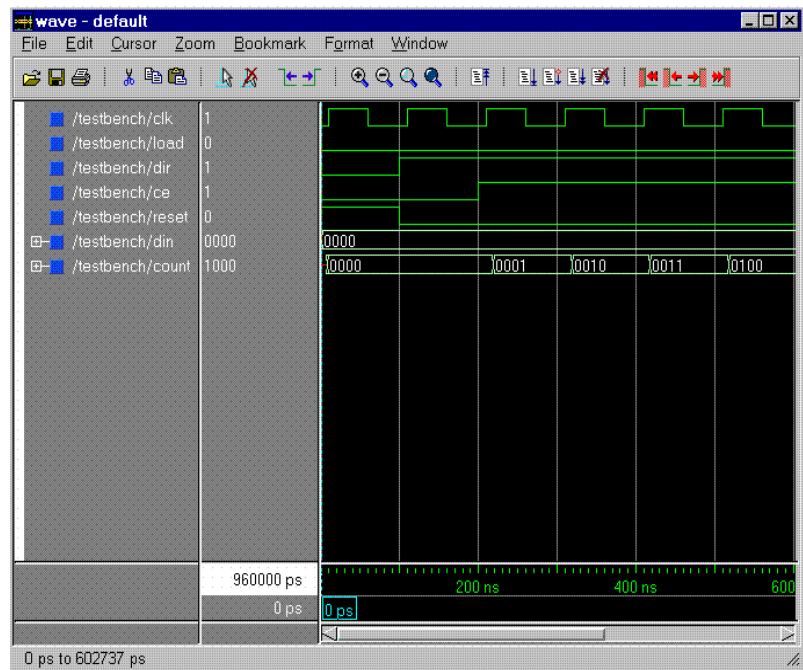


4. Click **Zoom** → **Zoom In** or click the Zoom In icon in the toolbar.



5. Scroll to the far left of the waveform.

The output waveform looks like that in [Figure 1-8](#).



**Figure 1-8 Post-place and Route Simulation Waveform**

6. Exit ModelSim by closing the main ModelSim window.

## Design Entry (Top-Level 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.

### Creating a Schematic Symbol for the VHDL Module

To create a schematic symbol for the VHDL module:

1. In the Sources in Project window, select your counter module, counter.vhd.
2. In the Processes for Current 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:

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.

### Instantiating VHDL Modules

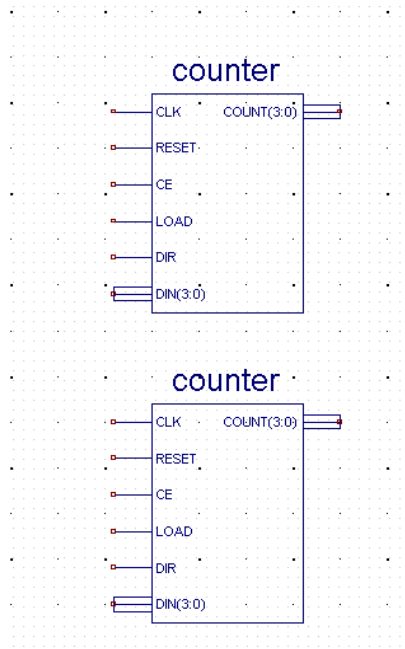
In ECS, instantiate two VHDL counter modules into the top-level schematic.

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



2. Select counter from the Symbols list in the Symbol browser (to the right of the screen). Do not select any options from 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.

Your schematic should look like the following diagram.



**Figure 1-9 Instantiated VHDL Modules**

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.

## Wiring the Schematic

When wiring the schematic symbols, some wires will be left hanging while others will interconnect the modules.

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

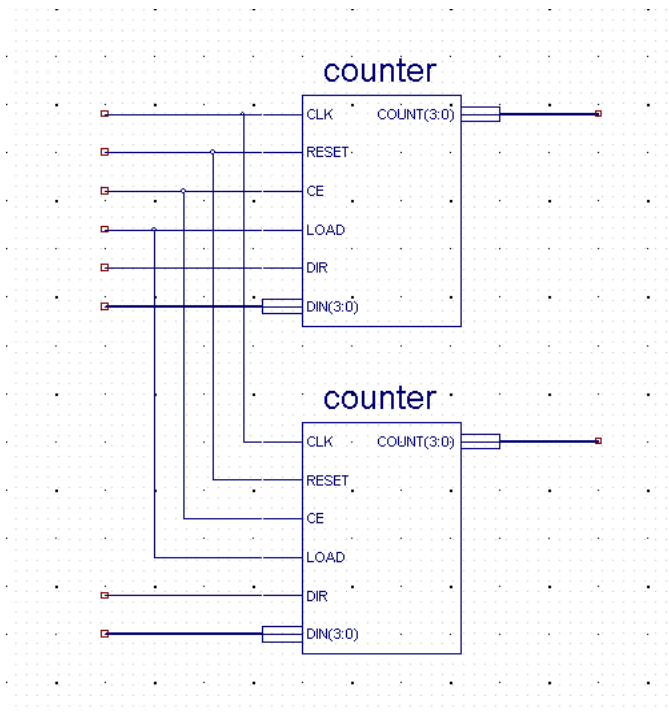


2. To add a hanging wire or to extend a wire:
  - a) Click once at the vertex of a pin on the first counter module.
  - b) Extend the wire to the desired length.
  - c) Double-click the location you want the wire to terminate.

**Note** Add a hanging wire to each module pin, according to the diagram [Figure 1-10](#).

3. To connect the wires of the two schematic symbols:
  - 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.

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



**Figure 1-10 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.



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 right side of the toolbar.

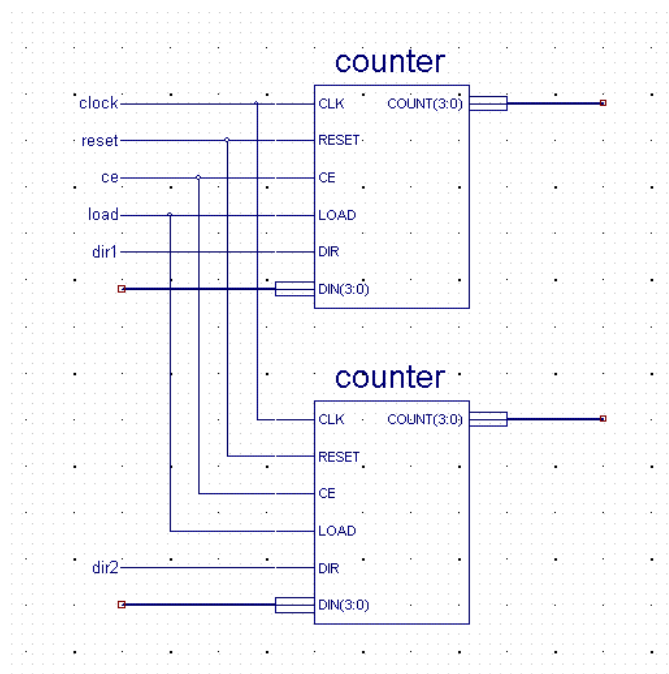


**Figure 1-11 Entering the net name**

**Note** Leave the default options Name Branch and Keep 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 1-12 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.



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 in the right side of the toolbar; for example, din1(3:0).

**Note** Leave the default options Name Branch and Keep 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.

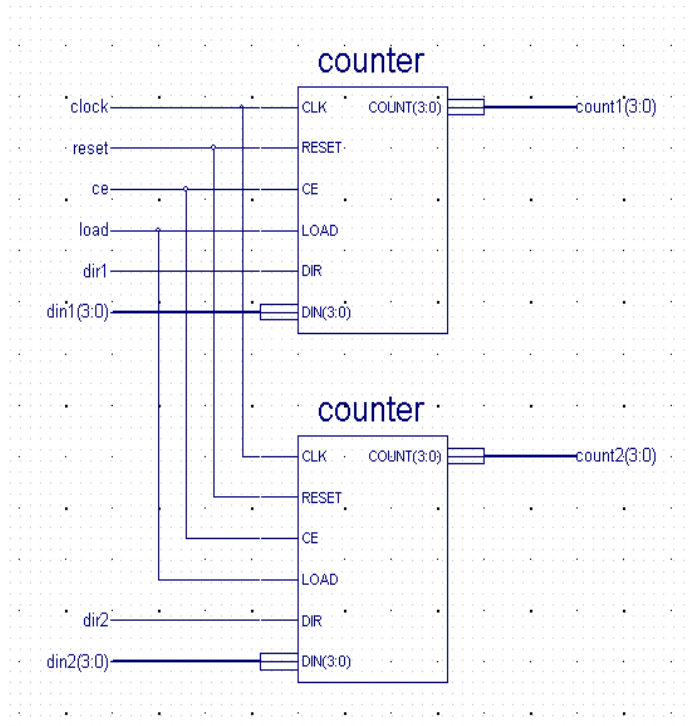


Figure 1-13 Schematic with Buses Added

## Adding I/O Markers

Next, identify the direction 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 1-14](#).

To add I/O markers:

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



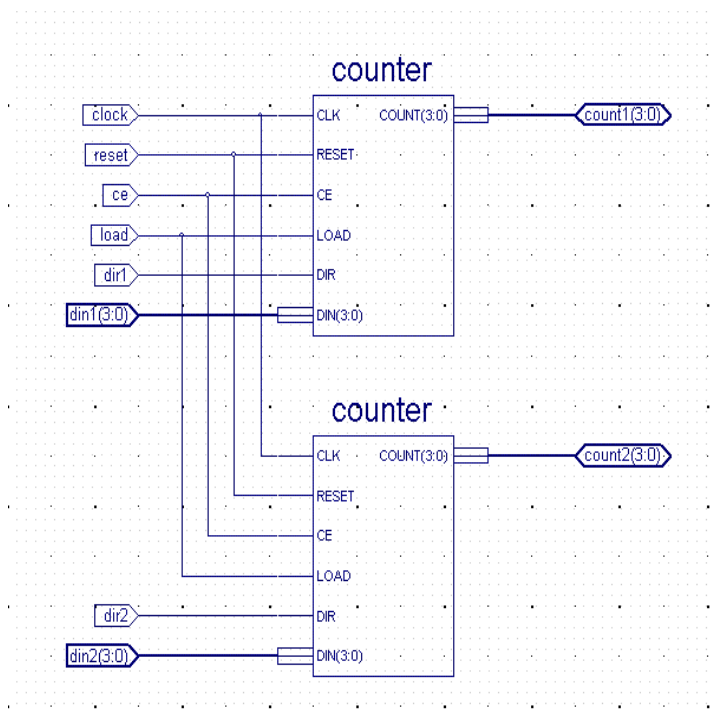
2. Add input markers to the clock, reset, ce, load, dir1 and dir2 wires, and the din1(3:0) and din2(3:0) buses as follows:
  - a) Select the Input type radio button from the Options toolbar.
  - b) Place the cursor, which now displays the input graphic, at the end of the counter input wire.
  - c) Click 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. When an attempt to add a marker is unsuccessful, an error message box appears.

3. Add bidirectional markers to the count wires as follows:
  - a) Select the Bidirectional radio button from the Options toolbar.
  - b) Click 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 1-14 Completed Schematic**

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

## Design Implementation

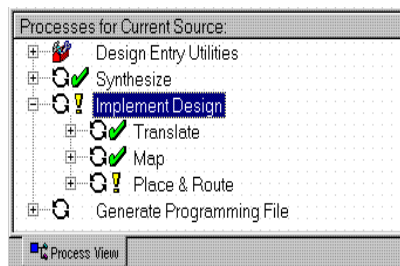
For this tutorial, design implementation covers two tasks: running the Implement Design process in Project Navigator, and viewing the resultant placed and routed design in Floorplanner.

### 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 Current Source window.

This runs all processes.



**Figure 1-15 Implement Design processes**

A check mark in the Processes for Current Source 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 the Floorplanner.

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

Floorplanner 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 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.

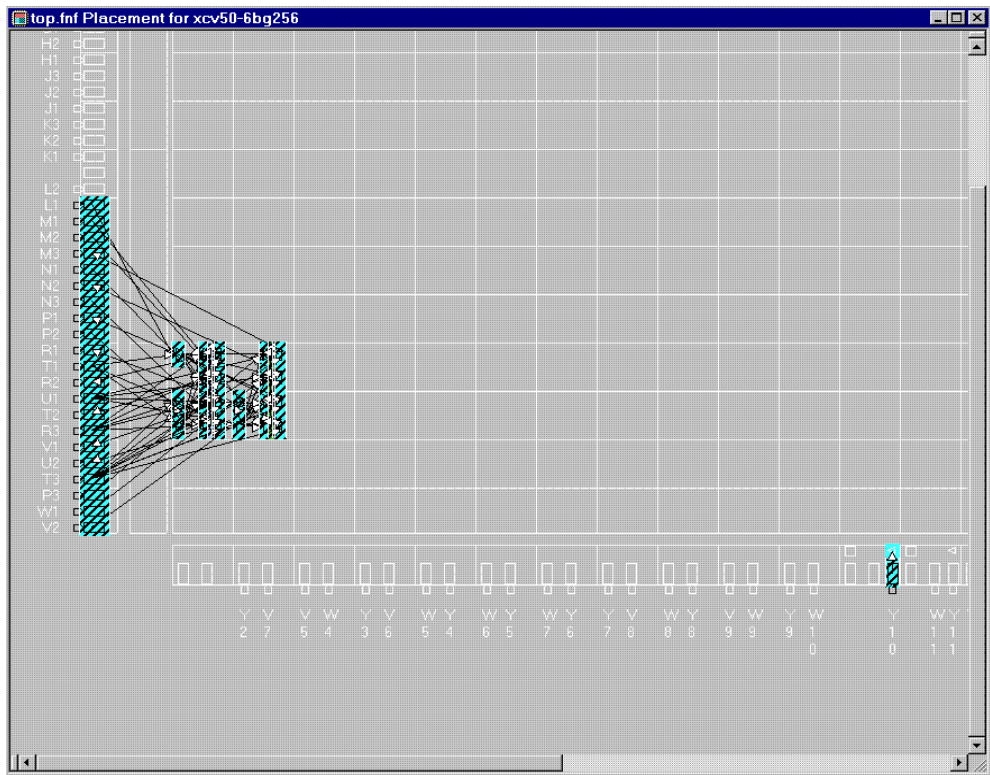
2. Select **View** → **Zoom** → **To Selected** or click the Zoom to Selected icon (the last icon on the far right of the toolbar) in the Floorplanner toolbar.



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 1-16 I/O Connections in Floorplanner

When you have finished viewing the implemented design, save the Floorplanner design view using **File** → **Save** and exit Floorplanner.

## Simulating the Top-level Design

Next, run a timing simulation on the top-level design created in the previous sections. First, create a testbench waveform for the top-level design using HDL Benchner, and then simulate the top-level design using ModelSim.

### Creating a Testbench Waveform Source

Create a testbench waveform which you will modify in HDL Benchner.

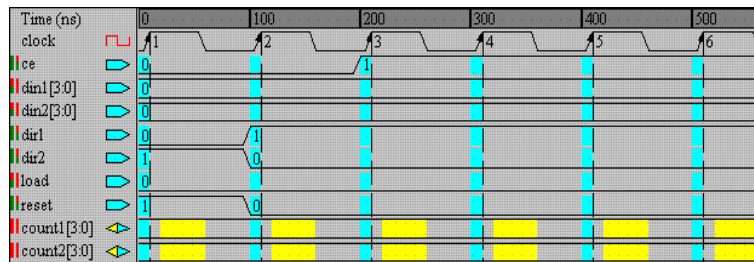
1. In Project Navigator, select top (top.sch) in the Sources in Project window.
2. Select **Project** → **New Source**.
3. In the New dialog box, select Test Bench Waveform source type.
4. Type the name 'top\_tbw'.
5. Select **Next**.
6. Ensure top is the associated source and select **Next**.
7. Select **Finish**.  
HDL Benchner is launched.
8. Click **OK** to use the default timing constraints for the testbench waveform.

### Initializing Counter Inputs

In the waveform in HDL Benchner, initialize the counter inputs as follows. Verify your entries using [Figure 1-17](#).

**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.

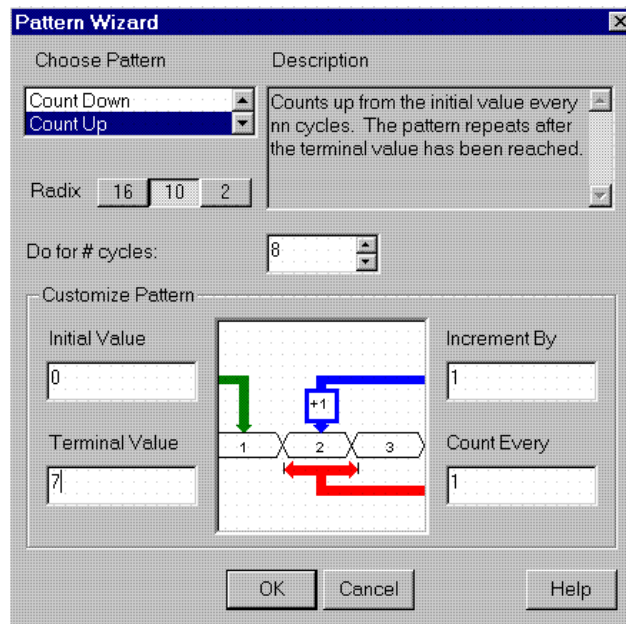


**Figure 1-17 HDL Benchner Stimulus and Response Entries**

## Generating the Expected Responses

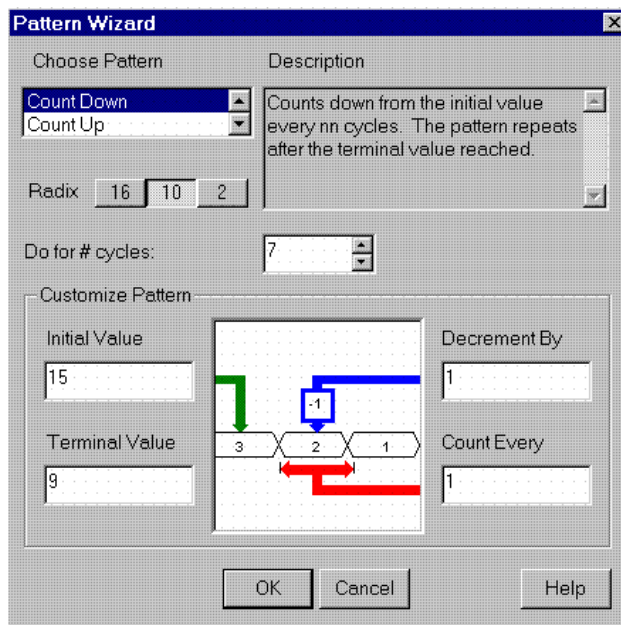
To generate the expected response, make the following response entries in the yellow areas in the waveform in HDL Benchner.

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:



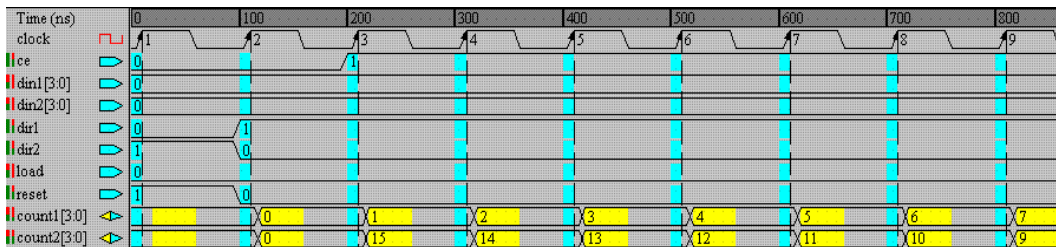
**Figure 1-18 Pattern Wizard Settings**

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 1-19 Pattern Wizard Settings**

9. Click **OK** in the Pattern Wizard dialog box.
- The testbench waveform should look like the following.



**Figure 1-20 Testbench Waveform**

10. Save the testbench waveform by selecting **File** → **Save Waveform** or by clicking the Save Waveform icon in the toolbar.



11. Exit HDL Benchner.

In the Source for Project window in the Project Navigator, the new testbench 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 the Project Navigator, select top\_tbw.tbw in the Sources in Project window.
2. In the Processes for Current Source window, double-click Simulate Post-Place and Route VHDL Model found in the ModelSim Simulator hierarchy.

ModelSim is launched with the back-annotated design.

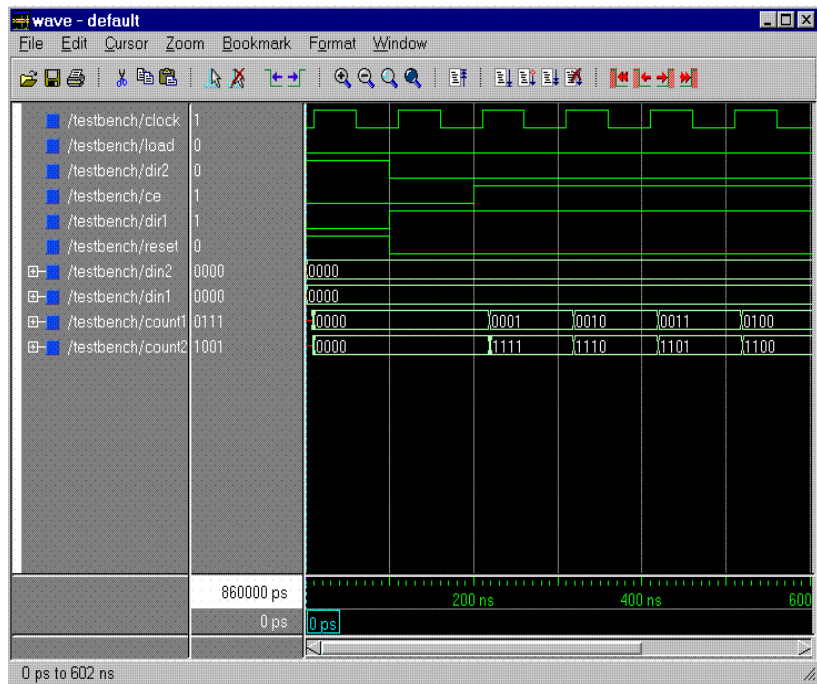
3. Click **Zoom** → **Zoom Full** or click the Zoom Full icon in the toolbar.



4. Click **Zoom** → **Zoom In** or click the Zoom In icon in the toolbar and scroll to the far left of the waveform.



The waveform should look like the following.



**Figure 1-21 Timing Simulation Waveform**

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

# Appendix: EDIF Design

---

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”

## Design Entry

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

### Creating a New Project

To create a new project:

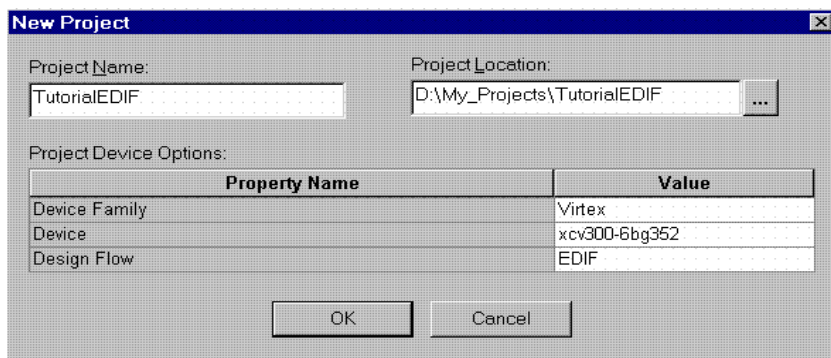
1. Select **File** → **New Project**.
2. In the New Project dialog box, type the desired location in the Project Location field, or browse to the directory under which you want to create your new project directory using the browse button next to the Project Location field.
3. Enter ‘TutorialEDIF’ in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

4. 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: xcv300-6bg352
  - ◆ Design Flow: EDIF
5. Click **OK**.



**Figure A-1 New Project Dialog Box**

ISE creates a subdirectory and a new project in Project Navigator.

## Adding the EDIF source file

Now, add an EDIF source file to the new project. 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:

1. Select **Project** → **Add Source**.
2. Browse to the following directories found in the Xilinx install directory:  
/ISEexamples/edif\_flow/.
3. Select the file 'mf.edn'.
4. Click **Open**.

The file is now added to the new project.

**Note** To add a copy of the EDIF netlist in the new project directory rather than the original file, select the **Project** → **Add Copy of Source** option instead of **Project** → **Add Source**.

## Design Implementation

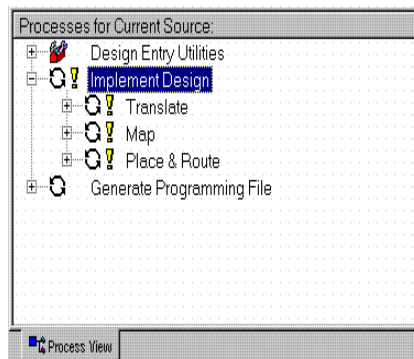
Next, implement the design and use FPGA Editor to view the placed and routed design.

### Running Implement Design

To run design implementation on the design:

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

This runs all processes (Translate through Place & Route) required to view the implemented design in FPGA Editor.



**Figure A-2 Implement Design processes**

A check mark in the Processes for Current Source 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

To view the design in FPGA Editor:

1. When implementation is finished, in the Processes for Current Source window, click the + sign beside Implement Design and click the + sign beside Place & Route.

2. Double-click the 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 in the Type column.

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

	Site	Type	#Pins	Hilited	Selected
1	GCLKBUF1	GCLK	2		No
2	AF14	GCLKIOB	1		No
3	A16	IOB	3		No
4	AD25	IOB	1		No
5	T24	IOB	1		No
6	T26	IOB	1		No
7	AE21	IOB	1		No
8	AD20	IOB	1		No
9	AF21	IOB	1		No
10	D15	IOB	1		No
11	C16	IOB	1		No

**Figure A-3 Listing all components by type**

When finished viewing the design, exit FPGA Editor.

# Index

---

## B

- behavioral simulation
  - with HDL Benchner, 1-11, 1-29
  - with ModelSim, 1-12
- buses, creating in schematics, 1-21

## C

- comment lines, 1-7
- constraints
  - timing, 1-9, 1-28

## D

- design entry
  - EDIF netlist, A-1
  - schematics, 1-16
  - VHDL counter module, 1-4
- design implementation
  - example, 1-25, A-3

## E

- EDIF
  - source file, A-2

## F

- Floorplanner
  - viewing design, 1-26
- FPGA Editor
  - viewing design, A-4

## H

- HDL Benchner
  - creating waveform, 1-9, 1-28

## I

- I/O markers, adding, 1-22
- instantiation
  - VHDL modules, 1-16
- ISE software, starting, 1-3

## L

- Language Templates, 1-6

## M

- markers, adding, 1-22
- ModelSim, simulation, 1-12, 1-31

## N

- net names
  - adding to wires, 1-19
- new project, creating
  - EDIF design flow, A-1
  - VHDL design flow, 1-4

## O

- online help, 1-3

## **P**

Project Location field, 1-4, A-1

Project Name field, 1-4, A-1

Property Name field, 1-4, A-2

## **S**

schematic symbols, creating, 1-16

schematics

- adding I/O markers, 1-22

- creating buses, 1-21

- instantiating VHDL modules, 1-16

- wiring, 1-18

simulation

- behavioral, 1-11, 1-12, 1-29

- timing, 1-14, 1-31

starting, ISE software, 1-3

## **T**

testbench. See waveform

timing constraints, 1-9

timing simulation, 1-14, 1-31

top-level schematics, 1-16

## **V**

VHDL modules

- creating, 1-5

- creating schematic symbols, 1-16

- instantiating in schematics, 1-16

## **W**

waveform

- creating, 1-9, 1-28

wires

- in schematics, 1-18