ISE Quick Start Tutorial

XILINX®

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. 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. This tutorial is current for ISE 7.x.

Additional Resources

For additional information, go to http://www.xilinx.com/support. 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 http://www.xilinx.com/support/techsup/tutorials/
Answer Browser	Database of Xilinx solution records http://www.xilinx.com/xlnx/xil_ans_browser.jsp
Application Notes	Descriptions of device-specific design techniques and approaches http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=Application+Notes
Data Sheets	Device-specific information on Xilinx device characteristics, including readback, boundary scan, configuration, length count, and debugging http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp
Problem Solvers	Interactive tools that allow you to troubleshoot your design issues http://www.xilinx.com/support/troubleshoot/psolvers.htm
Tech Tips	Latest news, design tips, and patch information for the Xilinx design environment http://www.xilinx.com/xlnx/xweb/xil_tx_home.jsp

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`
`Courier bold`	Literal commands that you enter in a syntactical statement	**`ngdbuild`** *`design_name`*
Helvetica bold	Commands that you select from a menu	**File → Open**
	Keyboard shortcuts	**Ctrl+C**
Italic font	Variables in a syntax statement for which you must supply values	**`ngdbuild`** *`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 [ ]	An optional entry or parameter. However, in bus specifications, such as **`bus[7:0]`**, they are required.	**`ngdbuild`** [*`option_name`*] *`design_name`*
Braces { }	A list of items from which you must choose one or more	**`lowpwr ={on|off}`**
Vertical bar |	Separates items in a list of choices	**`lowpwr ={on|off}`**

Vertical ellipsis
.
.
.
	Repetitive material that has been omitted	`IOB #1: Name = QOUT'`
		`IOB #2: Name = CLKIN'`
		.
		.
		.

Horizontal ellipsis ...	Repetitive material that has been omitted	**`allow block`** *`block_name loc1 loc2 ... locn;`*

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 "Additional Resources" for details. Refer to "Title Formats" in Chapter 1 for details.
Red text	Cross-reference link to a location in another document	See Figure 2-5 in the *Virtex-II Handbook*.
Blue, underlined text	Hyperlink to a website (URL)	Go to http://www.xilinx.com for the latest speed files.

Table of Contents

ISE Quick Start Tutorial

In this tutorial, you will create a new project in which you will design a 4-bit counter module, add constraints, simulate and implement the design, and view the results. Finally, you will generate a bitstream and configure the device.

Note: This tutorial is designed for ISE 7.x, Windows version.

This tutorial contains the following sections:

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

Tutorial Overview

When you complete the tutorial you will know how to:

- Create an ISE project for a Spartan-3 FPGA device.
- Create a top-level HDL design and verify that your HDL code uses the correct syntax.
- Create a test bench waveform to be used in simulation of the design.
- Create a User Constraints File.
- Apply timing and pin location constraints to the design.
- Perform behavioral and post-place and route simulations on the design.
- Synthesize and implement your design.
- Verify that constraints were applied to your design and timing is met.
- View the placed and routed design in FPGA Editor.
- View the area groups of the design in Floorplanner.
- Create a configuration bitstream.

Getting Started

Software Requirements

To use this tutorial, you must install the following software:

- ISE 7.x.

  For more information about installing Xilinx® software, see the *ISE Release Notes and Installation Guide* at: http://www.xilinx.com/support/software_manuals.htm.

- If you are running the WebPack configuration, you need a licensed ModelSim™ simulator that supports either VHDL or Verilog HDL simulation. You may wish to install the Starter Version of MXE (ModelSim Xilinx Edition).

  For more information about ModelSim simulators, please refer to the Answer Browser, and see Answer 9859.

Starting the ISE Software

For Windows users, start ISE from the Start menu by selecting:

Start → Programs → Xilinx ISE 7 → Project Navigator

The ISE Project Navigator opens. The Project Navigator lets you manage the sources and processes in your ISE project. All of the tasks in the Quick Start Tutorial are managed from within Project Navigator.

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

Stopping and Restarting a Session

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

To stop the session:

- Save all source files you have opened in other applications.
- Exit the software (ISE and other applications).

The current status of the ISE project is maintained when exiting the software.

To restart your session, start the ISE software again. ISE displays the contents and state of your project with the last saved changes.

Accessing Help

At any time during the tutorial, you can access online help for additional information about a variety of topics and procedures in the ISE software as well as related tools.

To open Help you may do either of the following:

- Press **F1** to view Help for the specific tool or function that you have selected or highlighted.

- Launch the **ISE Help Contents** from the Help menu. It contains information about creating and maintaining your complete design flow in ISE.

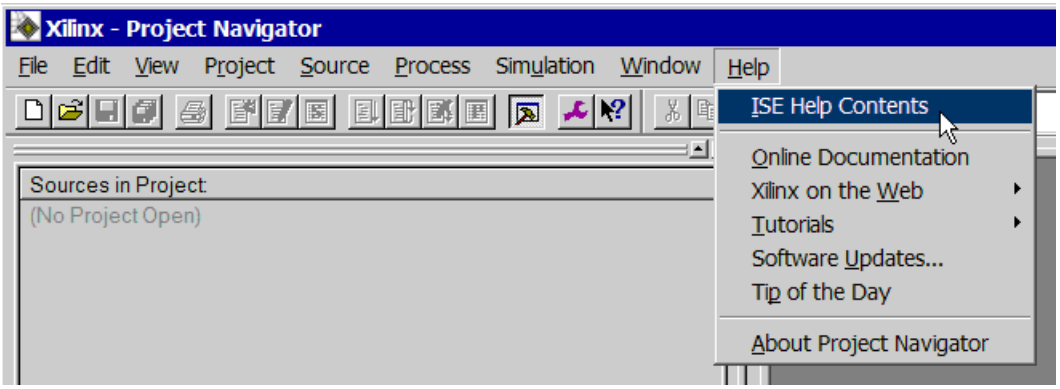

Figure 1: **Launching ISE Help Contents**

Creating a New Project in ISE

In this section, you will create a new ISE project. A project is a collection of all files necessary to create and to download a design to a selected FPGA or CPLD device.

To create a new project for this tutorial:

1. Select **File** > **New Project**. The New Project Wizard appears.
2. First, enter a location (directory path) for the new project.
3. Type **tutorial** in the Project Name field.

When you type **tutorial** in the Project Name field, a tutorial subdirectory is created automatically in the directory path you selected.

4. Select **HDL** from the Top-Level Module Type list, indicating that the top-level file in your project will be HDL, rather than Schematic or EDIF.
5. Click **Next** to move to the project properties page.
6. Fill in the properties in the table as shown below

   Device Family: **Spartan3**

   Device: **xc3s200**

   Package: **ft256**

   Speed Grade: **-4**

   Top-Level Module Type: **HDL**

   Synthesis Tool: **XST**

   Simulator: **ISE Simulator** (or **ModelSim**)

   Generated Simulation Language: **VHDL** or **Verilog**, depending on the language you want to use when running behavioral simulation.

When the table is complete, your project properties should look like the following:

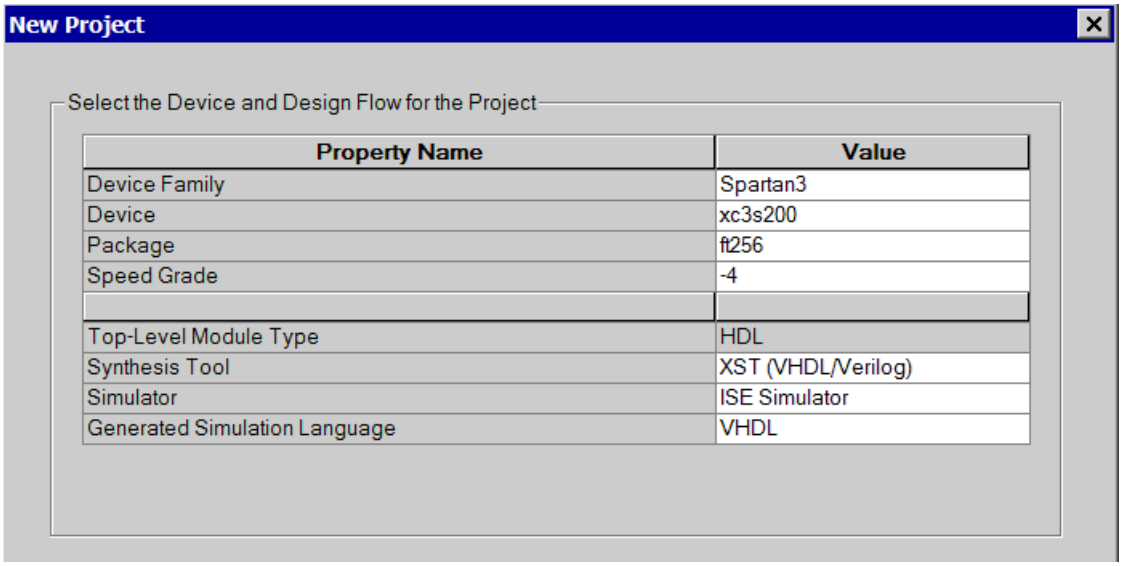

Figure 2: **Project Properties**

7. Click **Next** to proceed to the Create New Source window in the New Project Wizard. At the end of the next section, your new project will be created.

Creating an HDL Source

In this section, you will create a top-level HDL file for your design. Determine the language that you wish to use for the tutorial. Then, continue either to the "Creating a VHDL Source" section below, or skip to the "Creating a Verilog Source" section.

Note: When you create a new project using the New Project Wizard, you can only create *one* new source. Additional HDL or schematic sources can be created from within ISE, after your new project has been created. For an in-depth explanation of schematics, see the ISE In-Depth Tutorial at http://www.xilinx.com/support/techsup/tutorials.

Creating a VHDL Source

This simple counter design has two inputs: CLOCK and DIRECTION. The direction of the up-down counter is indicated by the DIRECTION input. This design has one 4-bit bus output called COUNT_OUT.

1. Click **New Source** in the New Project Wizard to add one new source to your project.
2. Select **VHDL Module** as the source type in the New Source dialog box.
3. Type in the file name **counter**.
4. Verify that the **Add to project** checkbox is selected.
5. Click **Next**.
6. Define the ports for your VHDL source.

   In the Port Name column, type the port names on three separate rows: **CLOCK**, **DIRECTION**, and **COUNT_OUT**.

In the Direction column, indicate whether each port is an input, output, or inout. For CLOCK and DIRECTION, select **in** from the list. For COUNT_OUT, select **out** from the list.

To indicate that COUNT_OUT is a 4-bit bus, use the arrows to select **3** in the MSB (Most Significant Bit) field, and select **0** in the LSB (Least Significant Bit) field.

7. Click **Next** in the Define VHDL Source dialog box.
8. Click **Finish** in the New Source Information dialog box to complete the new source file template.
9. Click **Next** in the New Project Wizard.
10. Click **Next** again.
11. Click **Finish** in the New Project Information dialog box.

ISE creates and displays the new project in the Sources in Project window and adds the counter.vhd file to the project.

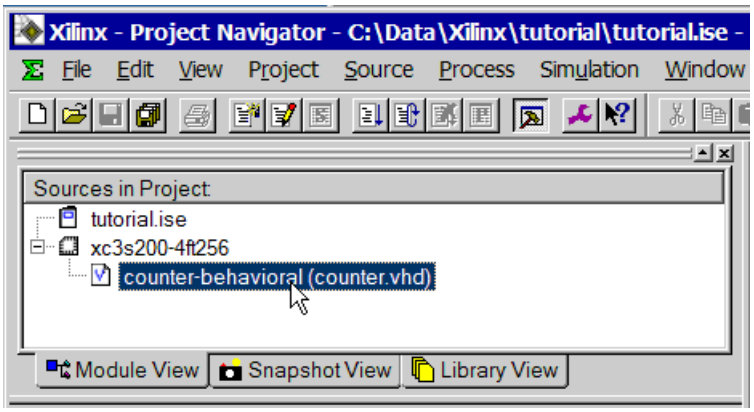

Figure 3: **Sources in Project**

12. Double-click on the **counter.vhd** file in the Sources in Project window to open the VHDL file in the ISE Text Editor.

The counter.vhd file contains:

- Header information.
- Library declaration and use statements.
- Entity declaration for the counter and an empty architecture statement.

13. In the header section, fill in the following fields:

    Design Name: **counter.vhd**

    Project Name: **counter**

    Target Device: **xc3s200-4ft256**

    Description: **This is the top level HDL file for an up/down counter.**

    Dependencies: **None**

Note: It is good design practice to fill in the header section in all source files.

Using Language Templates (VHDL)

ISE provides numerous Language Templates to simplify writing HDL code. In this tutorial you will use an "inference" template for an Up / Down Simple Counter.

1. To open the templates, select **Edit → Language Templates**.

The Language Templates enable you to access many different types and styles of HDL for use in your projects.

2. Using the "**+**" symbol, browse to the following folder:

   VHDL → Synthesis Constructs → Coding Examples → Counters → Binary → Up/Down Counters → Simple Counter

3. Cut and paste this code snippet from the Language Template to the counter.vhd file. Place it between the `begin` and `end` statements of the `architecture` section.
4. Replace the names shown within the angle brackets <> with the port names that you defined during creation of the VHDL code. Although VHDL is a case insensitive language, it is good design practice to maintain the correct case. The following replacements should be made:

   replace `<clock>` with `CLOCK`

   replace `<count_direction>` with `DIRECTION`

   replace `<count>` with `COUNT_OUT`

5. Close the Language Templates.

Final Editing of the VHDL Source

1. In VHDL, the output port cannot be read from within the design, therefore a new temporary signal must be made. Type the following line below the architecture declaration and above the first `begin` statement:

```
signal count_int : std_logic_vector(0 to 3) := "0000";
```

2. In the `process` statement, change the code to use `count_int` in the following places:

```
if DIRECTION ='1' then
            count_int <= count_int + 1;
         else
            count_int <= count_int - 1;
 end if;
```

3. Below the `end process` statement, enter the following line:

```
COUNT_OUT <= count_int;
```

4. Save the file by selecting **File → Save**.

When you are finished, the code for the counter should look like the following:

```
architecture Behavioral of counter is
signal count_int : std_logic_vector(0 to 3) := "0000";
begin
process (CLOCK)
begin
   if CLOCK='1' and CLOCK'event then
      if DIRECTION='1' then
         count_int <= count_int + 1;
      else
```

```
            count_int <= count_int - 1;
         end if;
      end if;
   end process;
   COUNT_OUT <= count_int;
   end Behavioral;
```

You have now created the VHDL source for the tutorial project. Skip past the Verilog sections below, and proceed to the "Checking the Syntax of the New Counter Module"section.

Creating a Verilog Source

This simple counter design has two inputs: CLOCK and DIRECTION. The direction of the up-down counter is indicated by the DIRECTION input. This design has one 4-bit bus output called COUNT_OUT.

1. Click **New Source** in the New Project dialog box to add one new source to your project.
2. Select **Verilog Module** as the source type in the New Source dialog box.
3. Type in the file name **counter**.
4. Verify that the **Add to Project** checkbox is selected.
5. Click **Next**.
6. Define the ports for your Verilog HDL source:

   In the Port Name column, type the port names on three separate rows: **CLOCK**, **DIRECTION**, and **COUNT_OUT**.

   In the Direction column, indicate whether each port is an input, output, or inout. For CLOCK and DIRECTION, select **input** from the list. For COUNT_OUT, select **output** from the list.

   To indicate that COUNT_OUT is a 4-bit bus, use the arrows to select **3** in the MSB (Most Significant Bit) field, and select **0** in the LSB (Least Significant Bit) field.
7. Click **Next** in the Define Verilog Source dialog box.
8. Click **Finish** in the New Source Information dialog box to complete the new source file template.
9. Click **Next** in the New Project Wizard.
10. Click **Next** again.
11. Click **Finish** in the New Project Information dialog box.

ISE creates and displays the new project in the Sources in Project window and adds the counter.v file to the project.

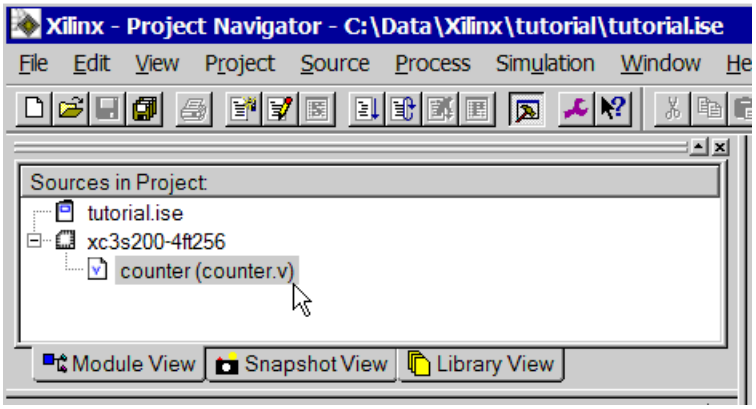

Figure 4: **Sources in Project**

12. Double-click on the **counter.v** file in the Sources in Project window to open the Verilog file in the ISE Text Editor.

The counter.v file contains:

- Header information.
- Library declaration and use statements.
- Entity declaration for the counter and an empty architecture statement.

13. In the header section, fill in the following fields:

    Design Name: **counter.v**

    Project Name: **counter**

    Target Device: **xc3s200-4ft256**

    Description: **This is the top level HDL file for an up/down counter**.

    Dependencies: **None**

Note: It is good design practice to fill in the header section in all source files.

Using Language Templates (Verilog)

ISE provides Language Templates to simplify writing HDL code. In this tutorial you will use an "inference" template for an Up / Down Simple Counter.

1. To open the templates, select **Edit** → **Language Templates**.

The Language Templates enable you to browse to many different types and styles of HDL for use in your projects.

2. For this tutorial, browse to

   Verilog → **Synthesis Constructs** → **Coding Examples** → **Counter** → **Binary** → **Up/Down Counters** → **Simple Counter**

3. Cut and paste this code snippet from the Language Template to the counter.v file. Place it between the last port declaration and the `endmodule` statement.
4. Replace the names shown within the angle brackets <> with the port names that you defined during creation of the Verilog code. Verilog is a case sensitive language, so the

proper case must be maintained as you edit the code. The following replacements should be made:

replace `<clock>` with `CLOCK`

replace `<reg_name>` with `count_int`

replace `<up_down>` with `DIRECTION`

5. Close the Language Templates.

Final Editing of the Verilog Source

1. The `count_int` register must be initialized. To do this, replace the following line:

   `reg [<upper>:0] count_int;`

   with: `reg [3:0] count_int = 0;`

2. The main output must be assigned with the value from the internal reg. To do this, add the following line just above the `endmodule` statement:

   ```
   assign COUNT_OUT = count_int;
   ```

3. Save the file by selecting **File** → **Save**.

When you are finished, the code for the counter should look like the following:

```
module counter(CLOCK, DIRECTION, COUNT_OUT);
input CLOCK;
input DIRECTION;
output [3:0] COUNT_OUT;

reg [3:0] count_int = 0;
  always @(posedge CLOCK)
    if (DIRECTION)
       count_int <= count_int + 1;
    else
       count_int <= count_int - 1;

assign COUNT_OUT = count_int;
endmodule
```

You have now created the Verilog source for the tutorial project.

Checking the Syntax of the New Counter Module

When the source files are complete, the next step is to check the syntax of the design. Syntax errors and typos can be found using this step.

1. Select the **counter** design source in the ISE Sources window to display the related processes in the Processes for Source window.
2. Click the "**+**" next to the Synthesize-XST process to expand the hierarchy.

3. Double-click the **Check Syntax** process.

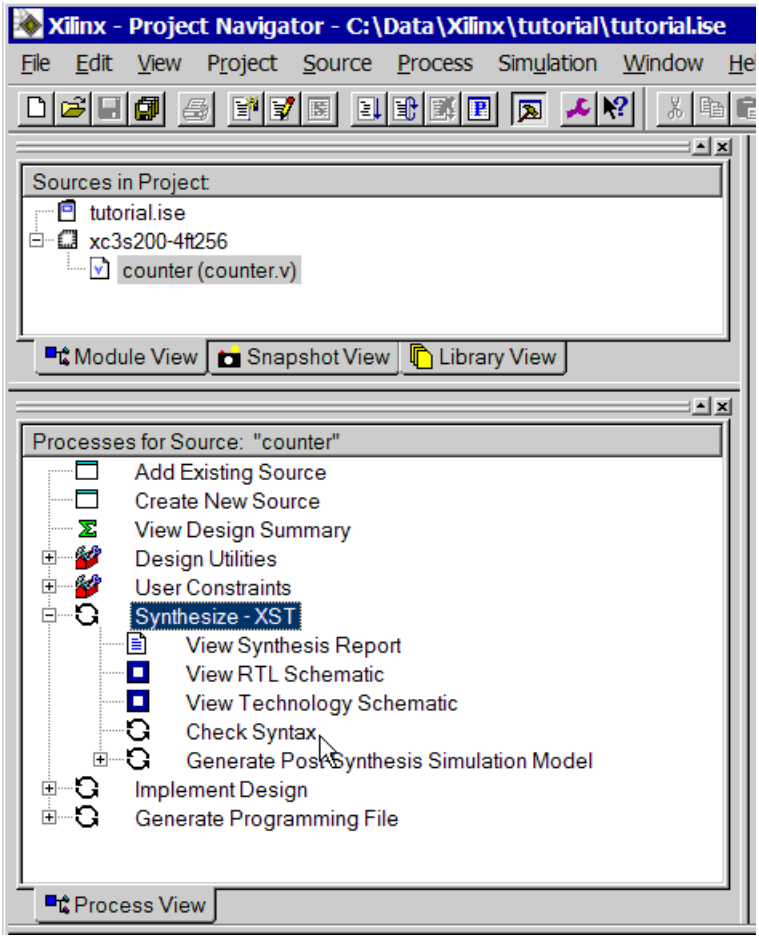

Figure 5: **Processed for Selected Source**

When an ISE process completes, you will see a status indicator next to the process name.

- If the process completed successfully, a green check mark appears.
- If there were errors and the process failed, a red X appears.
- A yellow exclamation point means that the process completed successfully, but some warnings occurred.
- An orange question mark means the process is out of date and should be run again.

4. Look in the **Console** tab of the Transcript window and read the output and status messages produced by any process that you run.

   Caution! You must correct any errors found in your source files. If you continue without valid syntax, you will not be able to simulate or synthesize your design.

Design Simulation

Creating a Test Bench for Simulation

In this section, you will create a test bench waveform containing input stimulus you can use to simulate the counter module. This test bench waveform is a graphical view of a test bench. It is used with a simulator to verify that the counter design meets both behavioral and timing design requirements. You will use the Waveform Editor to create a test bench waveform (TBW) file.

1. Select the **counter** HDL file in the Sources in Project window.
2. Create a new source by selecting **Project** → **New Source**.
3. In the New Source window, select **Test Bench Waveform** as the source type, and type **testbench** in the File Name field.
4. Click **Next**.
5. The Source File dialog box shows that you are associating the test bench with the source file: `counter`. Click **Next**.
6. Click **Finish**.

You need to set initial values for your test bench waveform in the Initialize Timing dialog box before the test bench waveform editing window opens.

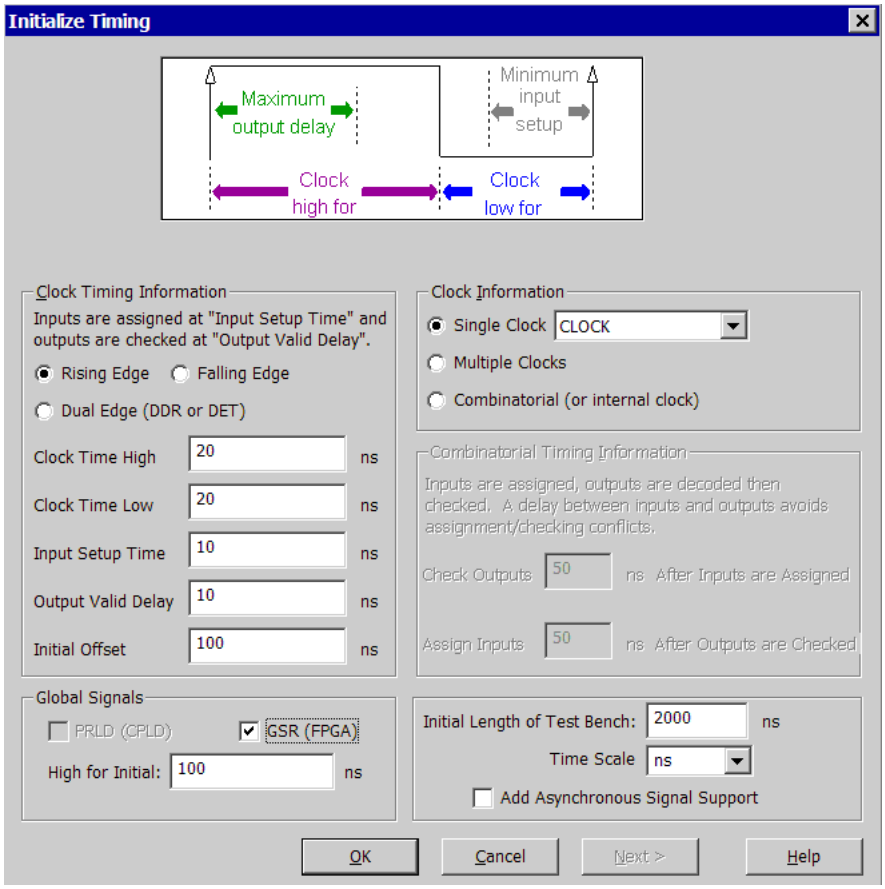

Figure 6: **Waveform Editor - Initialize Timing Dialog Box**

7. Fill in the fields in the Initialize Timing dialog box using the information below:
   - Clock Time High: **20** ns.
   - Clock Time Low: **20** ns.
   - Input Setup Time: **10** ns.
   - Output Valid Delay: **10** ns.
   - Initial Offset: **0** ns
   - Global Signals: **GSR (FPGA)**

   Note: The GSR value of 100 is added to the Initial Offset value automatically.

   - Initial Length of Test Bench: **2000** ns.

   Leave the remaining fields with their default values.
8. Click **OK** to open the waveform editor.

Note: You may find it easier to work on the test bench if you un-dock the Waveform Editor from ISE. Click on the "**>>**" button to undock the window. You can un-dock and re-dock any embedded window this way.

The blue shaded areas are associated with each input signal and correspond to the Input Setup Time in the Initialize Timing dialog box. In this tutorial, the input transitions occur at the edge of the blue cells located under each rising edge of the CLOCK input.

9. In this design, the only stimulus that you will provide is on the DIRECTION port. Make the transitions as shown below for the DIRECTION port:
   - Click on the blue cell at approximately the 300 ns clock transition. The signal switches to high at this point.
   - Click on the blue cell at approximately the 900 ns clock transition. The signal switches back to low.
   - Click on the blue cell at approximately the 1400 ns clock transition. The signal switches to high again.

Note: For more accurate alignment, you can click on the Single Marker symbol in the top left corner of the Workspace and place the marker in the test bench by clicking in the desired location.

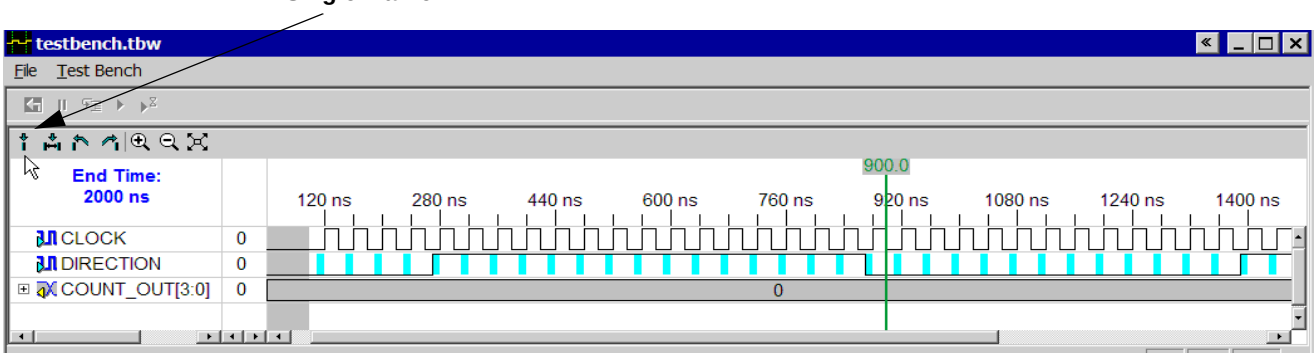

Figure 7: **Waveform Editor - Test Bench**

10. Select **File** → **Save** to save the waveform. In the Sources in Project window, the TBW file is automatically added to your project.
11. Close the Waveform Editor window.

Adding Expected Results to the Test Bench Waveform

In this step you will create a self-checking test bench with expected outputs that correspond to your inputs. The input setup and output delay numbers that were entered into the Initialize Timing dialog when you started the waveform editor are evaluated against actual results when the design is simulated. This can be useful in the Simulate Post-Place & Route HDL Model process, to verify that the design behaves as expected in the target device both in terms of functionality and timing.

To create a self-checking test bench, you can edit output transitions manually, or you can run the Generate Expected Results process:

1. Select the **testbench.tbw** file in the Sources in Project window.
2. Double-click the **Generate Expected Simulation Results** process. This process converts the TBW into HDL and then simulates it in a background process.
3. The **Expected Results** dialog box will open. Select **Yes** to post the results in the waveform editor.
4. Click the "**+**" to expand the COUNT_OUT bus and view the transitions that correspond to the Output Valid Delay time (yellow cells) in the Initialize Timing dialog box.

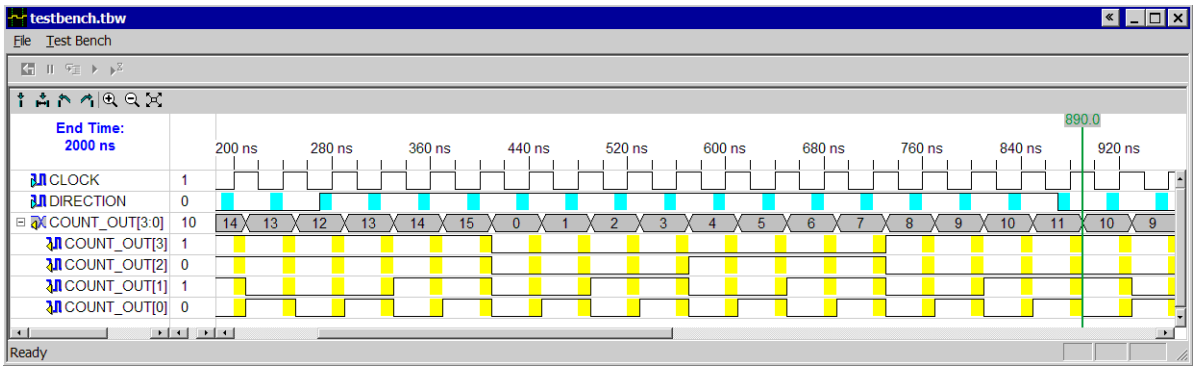

Figure 8: **Waveform Editor - Expected Results**

5. Select **File** → **Save** to save the waveform.
6. Close the Waveform Editor.

Now that you have a test bench, you are ready to simulate your design.

Simulating the Behavioral Model (ISE Simulator)

If you are using ISE Base or Foundation, you can simulate your design with the ISE Simulator. If you wish to simulate your design with a ModelSim simulator, skip this section and proceed to the "Simulating the Behavioral Model (ModelSim)" section.

To run the integrated simulation processes in ISE:

1. Select the **test bench** waveform in the Sources in Project window. You can see the Xilinx ISE Simulator processes in the Processes for Source window.

Note: You will only see ISE Simulator processes if you selected ISE Simulator as your project simulator.

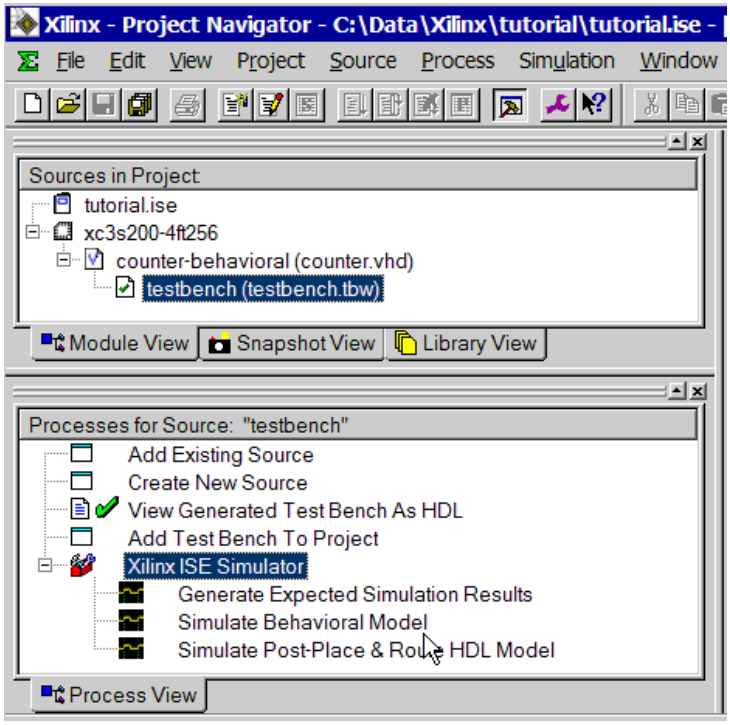

Figure 9: **Simulator Processes for Test Bench**

2. Double-click the **Simulate Behavioral Model** process.

The ISE Simulator opens and runs the simulation to the end of the test bench.

3. To see your simulation results, select the **testbench** tab and zoom in on the transitions. You can use the zoom icons in the waveform view, or right click and select a zoom command.

The ISE window, including the waveform view, should look like the following:

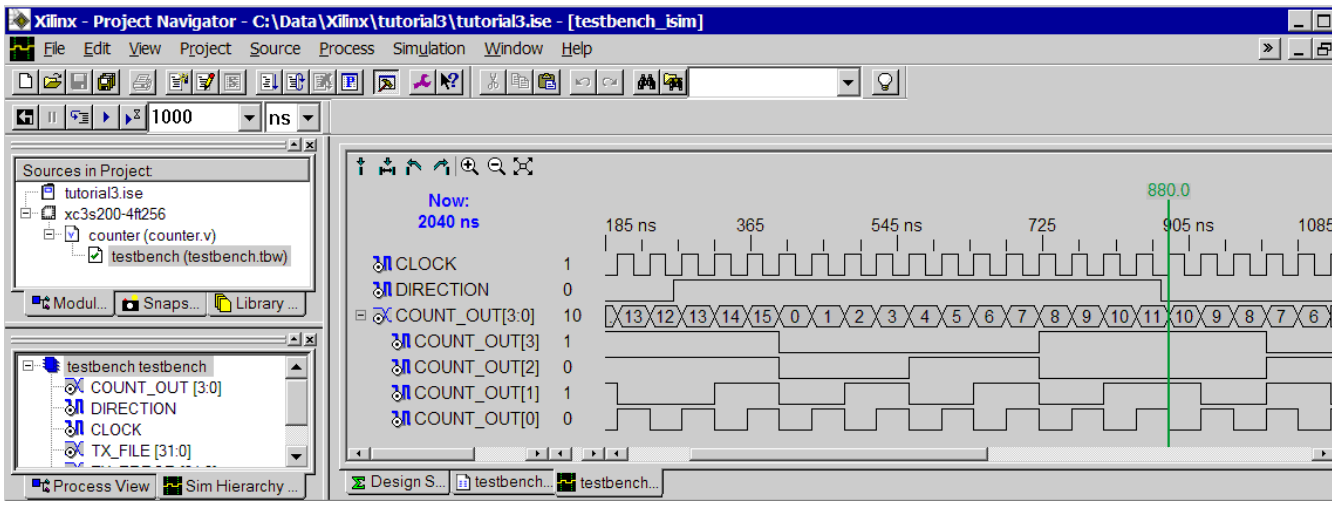

Figure 10: **Behavioral Simulation in ISE Simulator**

4. Zoom in on the area between 300 ns and 900 ns to verify that the counter is counting up and down as directed by the stimulus on the DIRECTION port.
5. Close the waveform view window.

You have completed simulation of your design using the ISE Simulator. Skip past the ModelSim section below and proceed to the "Creating and Editing Timing and Area Constraints"section.

Simulating the Behavioral Model (ModelSim)

If you have a ModelSim simulator installed, you can simulate your design using the integrated ModelSim flow. You can run processes from within ISE which launches the installed ModelSim simulator.

To run the integrated simulation processes in ISE:

1. Select the **test bench** in the Sources in Project window. You can see ModelSim Simulator processes in the Processes for Source window.

Note: You will only see ModelSim simulator processes if you selected ModelSim as your project simulator, and if your ModelSim simulator is properly installed. If the installation path for the ModelSim simulator is not found by ISE, you can specify it by selecting **Edit** → **Preferences** and entering your ModelSim path in the Model Tech Simulator field in the Integrated Tools page.

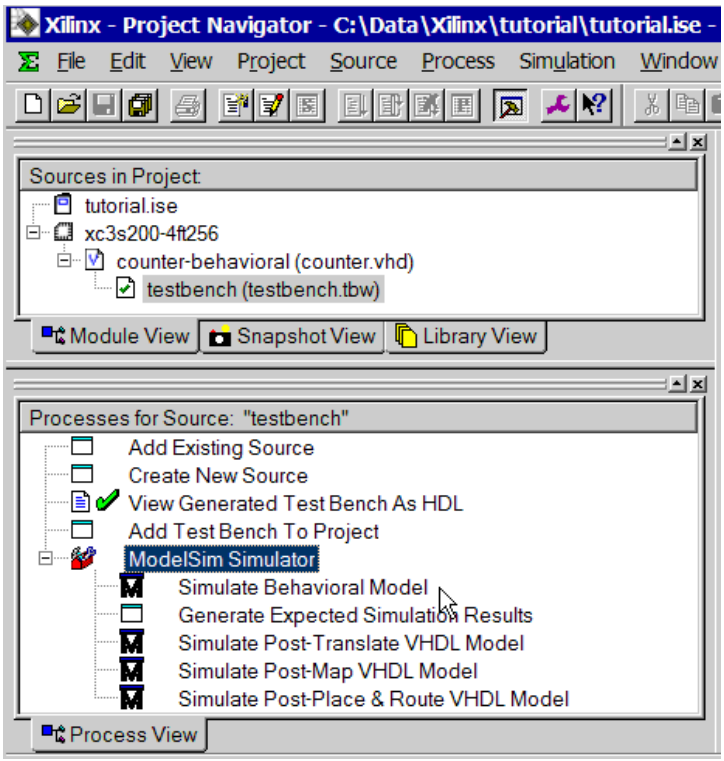

Figure 11: **Simulator Processes for Test Bench**

2. Double-click the **Simulate Behavioral Model** process.

The ModelSim simulator opens and runs your simulation to the end of the test bench.

The ModelSim window, including the waveform, should look like the following:

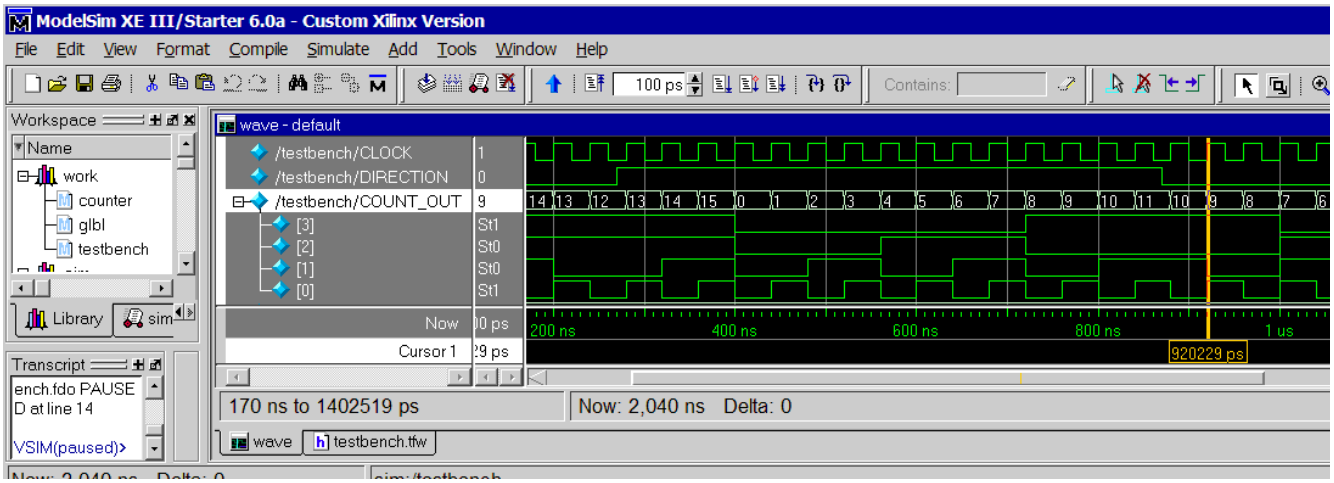

Figure 12: **Behavioral Simulation in ModelSim**

To see your simulation results, view the Wave window.

1. Right-click in the Wave window and select a zoom command.
2. Zoom in on the area between 300 ns and 900 ns to verify that the counter is counting up and down as directed by the stimulus on the DIRECTION port.
3. Close the ModelSim window.

Creating and Editing Timing and Area Constraints

In all designs, you will use constraints to assure that physical and timing requirements are met. There are several methods available for adding constraints to an ISE project.

Specifying Timing Constraints

As a minimum, every design should have a period constraint and offset constraints to achieve good performance results. In this section, you will add a constraints file with these timing constraints to your project.

1. Select the **counter** HDL source file in the Sources in Project window. The available processes are displayed in the Processes for Source window.
2. Click the "**+**" sign next to the **User Constraints** processes group. This expands the list of constraints processes.
3. Double-click the **Create Timing Constraints** process.

ISE runs the Synthesis and Translate steps and automatically creates a User Constraints File (UCF). You will be prompted with the following message:

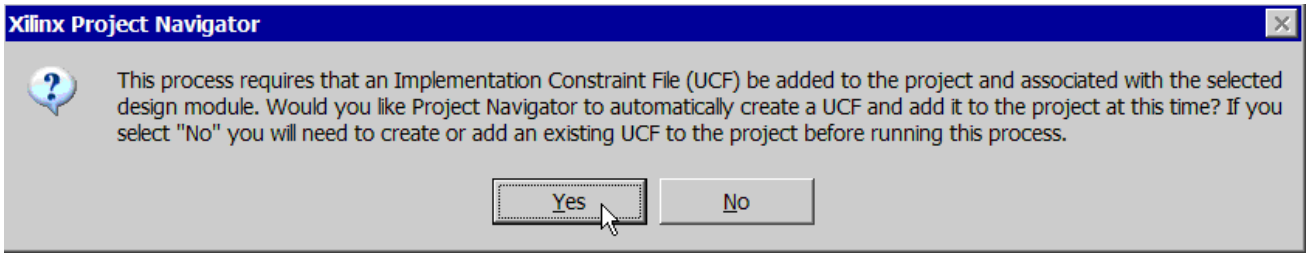

Figure 13: **Prompt to Add UCF file to Project**

4. Click **Yes** to add the UCF file to your project.

The `counter.ucf` file is added to your project and is visible in the Sources in Project window. The Xilinx Constraints Editor opens automatically.

Note: You can also create a UCF file for your project by selecting **Project** → **Create New Source**.

5. In the Constraints Editor Global tab, enter the following values in the fields associated with CLOCK. These values should match the period, input setup, and output delay times in your test bench waveform.
   - Period: **40** ns
   - Pad to Setup: **10** ns
   - Clock to Pad: **10** ns

As you complete each field, the constraint text as it will be written into the UCF file will appear in the lower window.

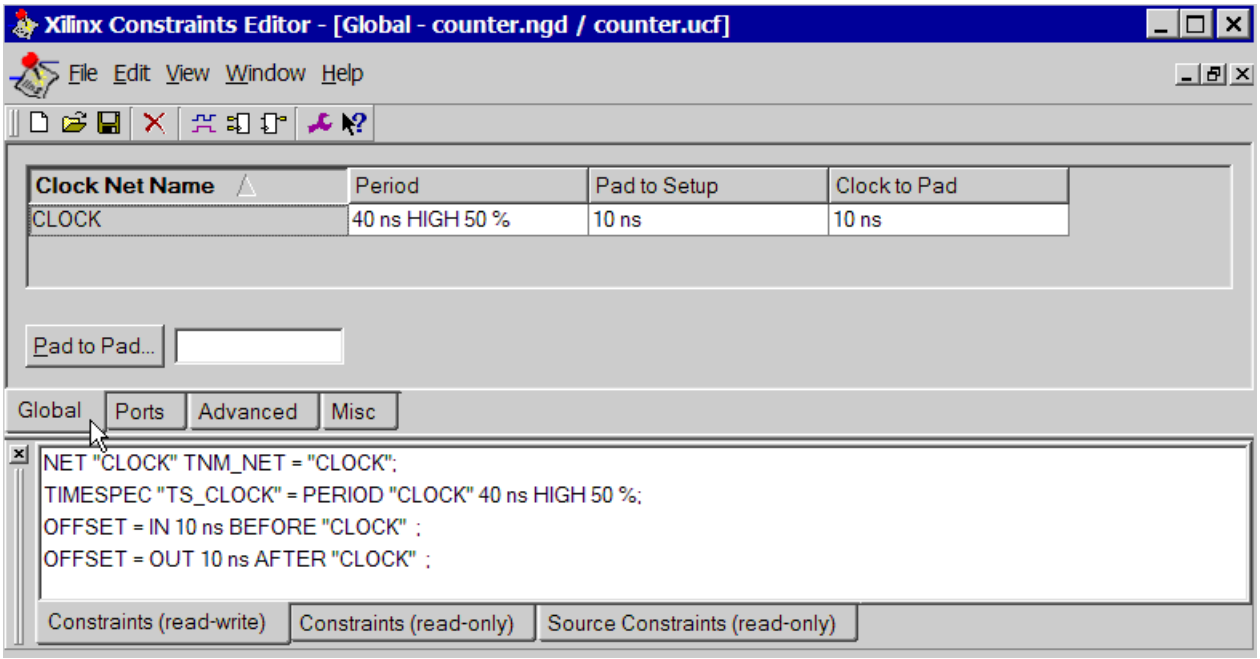

Figure 14: **Timing Constraints**

6. Save these timing constraints by selecting **File** → **Save**.
7. Close the Constraints Editor.

Assigning Pin Location

In addition to timing constraints, you need to add physical constraints to your design, even if only to associate certain pins on the device with specific inputs and outputs.

1. Double-click the **Assign Package Pins** process found in the User Constraints process group. The Xilinx Pinout and Area Constraints Editor (PACE) opens.
2. You can see your I/O Pins listed in the Design Object List window. Enter a pin location for each pin in the Loc column as specified below:
   - CLOCK: **T9**
   - COUNT_OUT<0>: **K12**
   - COUNT_OUT<1>: **P14**
   - COUNT_OUT<2>: **L12**
   - COUNT_OUT<3>: **N14**
   - DIRECTION: **K13**
3. Click on the **Package View** tab at the bottom of the window to see the pins you just added. Put your mouse over grid number K13 to verify that it is the DIRECTION pin.

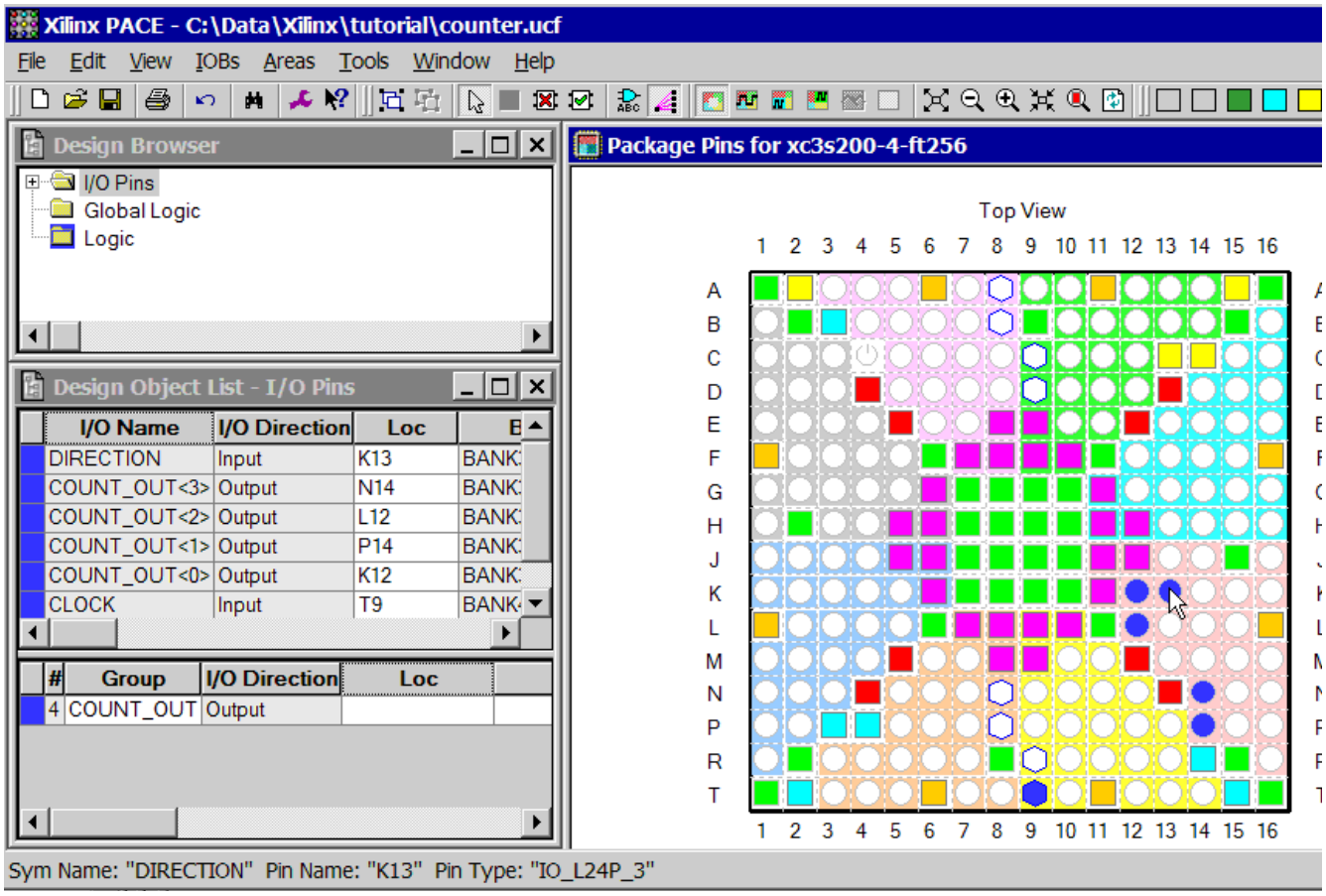

Figure 15: **View Pin Location Constraints in PACE**

4. Select **File** → **Save**. You are prompted to select the bus delimiter type based on the synthesis tool you are using. Select **XST Default <>** and click **OK**.
5. Close PACE.

Verify the UCF

The previous constraints processes have written constraints into the UCF file in your project. The UCF file is a text file, and you can open it in the ISE Text Editor.

1. Open the UCF file in text format by double-clicking the **Edit Constraints (Text)** process.
2. Verify the timing constraints (period and offsets) at the top of the file, and verify the PACE pin assignments in the PACE section.
3. Close the file using **File** → **Close**.

Design Synthesis and Implementation

Now that you have created the source files, verified the design's behavior with simulation, and added constraints, you are ready to synthesize and implement the design.

Your design will now go through the following steps:

- Synthesis with XST
- FPGA Implementation:
  - Translate (which runs the NGDBuild program)
  - Map
  - Place and Route (PAR).

Following PAR, you can do additional verification on your design before creating a configuration file for download to your FPGA.

Implementing the Design

1. Select the **counter** source file in the Sources in Project window.
2. In the Processes for Source window, click the "**+**" sign next to **Implement Design**. The Translate, Map, and Place & Route processes are displayed. Expand those processes as well by clicking on the "**+**" sign. You can see that there are many sub-processes and options that can be run during design implementation.
3. Double-click the top level **Implement Design** process.

ISE determines the current state of your design and runs the processes needed to pull your design through implementation. In this case, ISE runs the Translate, Map and PAR processes. Your design is now pulled through to a placed-and-routed state. This feature is called the "pull through model."

4. After the processes have finished running, notice the status markers in the Processes for Source window. You should see green checkmarks next to several of the processes, indicating that they ran successfully. If there are any yellow exclamation points, check the warnings in the Console tab or the Warnings tab within the Transcript window. If a red X appears next to a process, you must locate and fix the error before you can continue.

Verification of Synthesis

Your synthesized design can be viewed as a schematic in the Register Transfer Level (RTL) Viewer. The schematic view shows gates and elements independent of the targeted Xilinx® device.

1. In the Processes for Source window, double-click **View RTL Schematic** found in the Synthesize - XST process group. The top level schematic representation of your synthesized design opens in the workspace.
2. Right-click on the symbol and select **Push Into the Selected Instance** to view the schematic in detail.

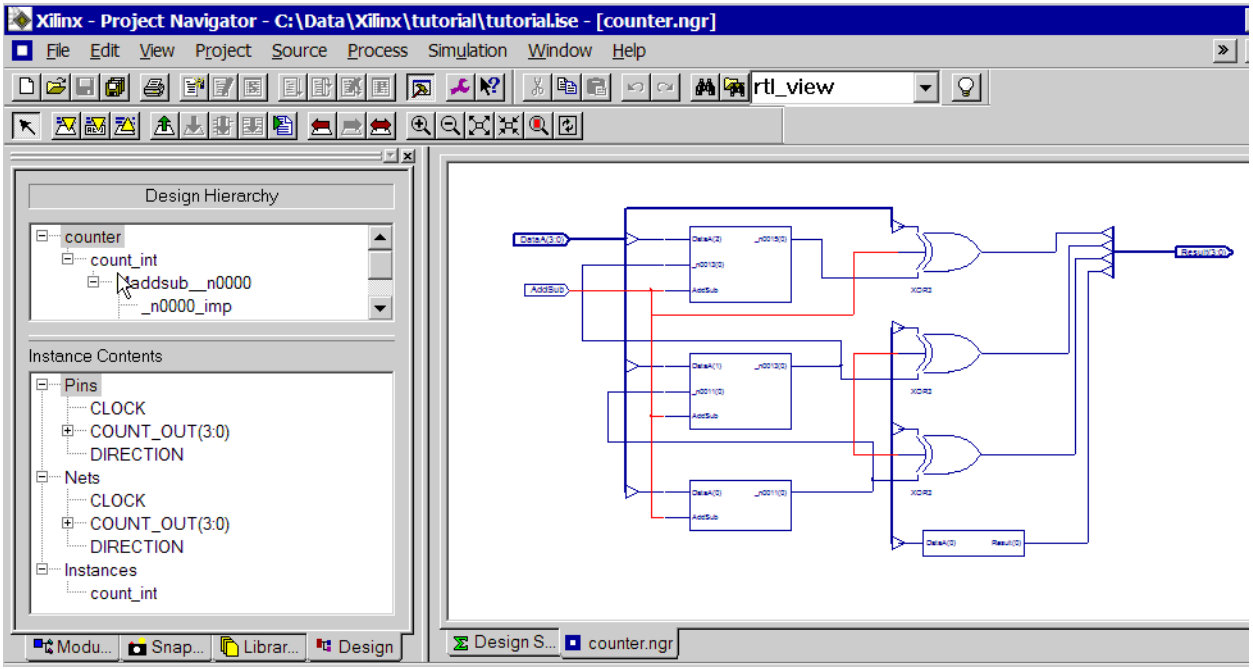

Figure 16: **RTL Viewer - Detailed View**

The Design tab appears in the Sources in Project window, enabling you to view the design hierarchy. In the schematic, you can see the design components you created in the HDL source, and you can “push into” symbols to view increasing levels of detail.

Note: You cannot edit this file. If you want to use a schematic as a source file in a design, you can create and edit a new schematic using the Schematic and Symbol Editors.

3. Close the schematic window.

Verification of the Implemented Design

After implementation is complete, you can verify your design before downloading it to a device.

Viewing Placement

In this section, you will use the Floorplanner to verify your pinouts and placement. Floorplanner is also very useful for creating area groups for designs.

1. Select the **counter** source file in the Sources in Project window.
2. Click the "**+**" sign to expand the **Place & Route** group of processes.
3. Double-click the **View/Edit Placed Design (Floorplanner)** process. The Floorplanner view opens.
4. Select **View** → **Zoom** → **ToBox** and then use the mouse to draw a box around the counter instance, shown in green on the right side of the chip.

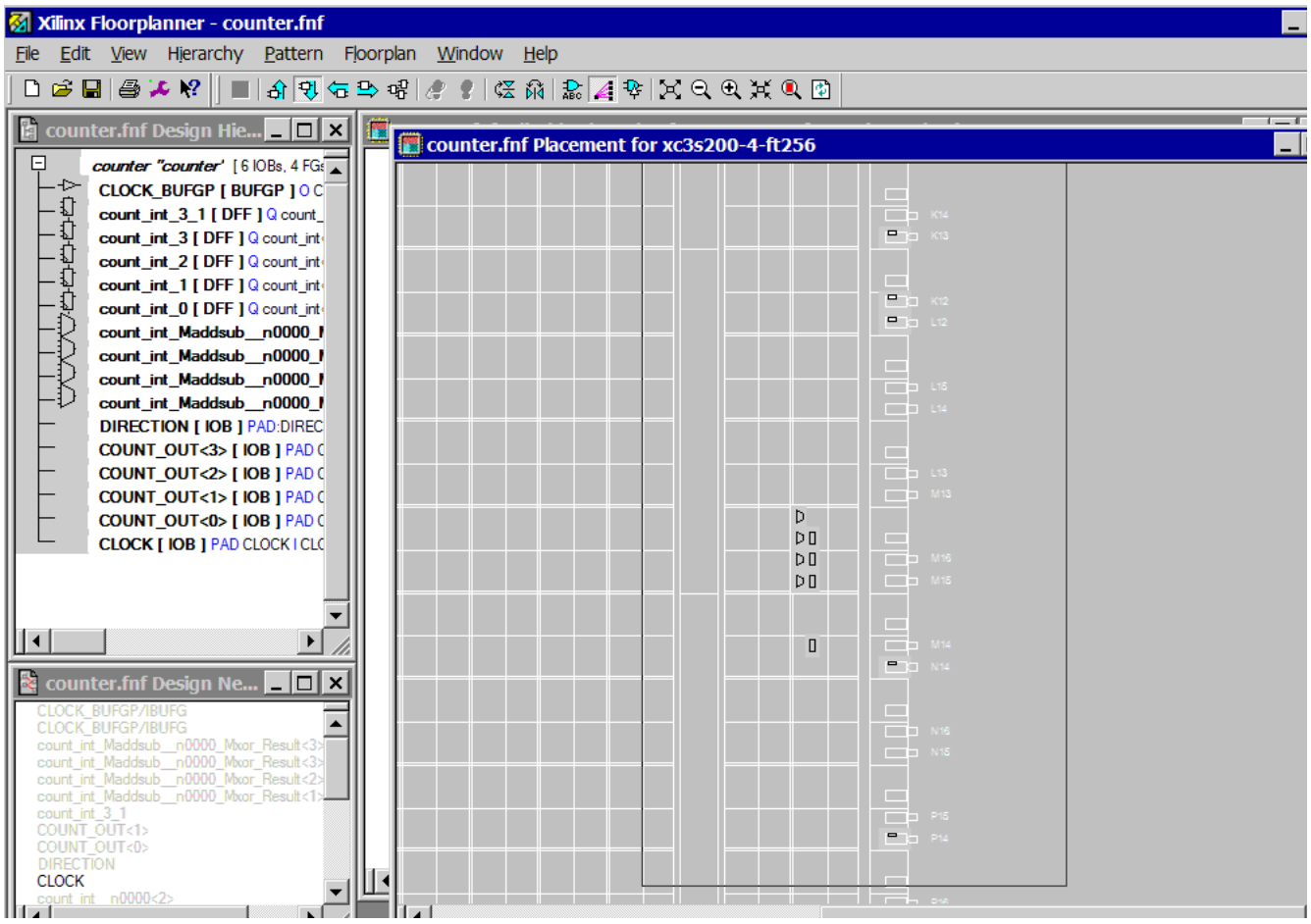

Figure 17: **Floorplanner View - Detailed View**

5. This view shows where the entire design was placed. Click on any of the components listed in the Design Hierarchy window to see where each component is placed.

6. Zoom in to the right side of the chip even more, and place your mouse over the K13 pad. You can see that your pinout constraint was applied - the DIRECTION pin is placed at K13.
7. Close the Floorplanner without saving.

Viewing Resource Utilization in Reports

Many ISE processes produce summary reports which enable you to check information about your design after each process is run. Detailed reports are available from the Processes for Source window. You can also view summary information and access most often-utilized reports in the Design Summary.

1. Click on the **Design Summary** tab at the bottom of the window. If you closed the summary during this tutorial, you can reopen it by double-clicking the **View Design Summary** process.

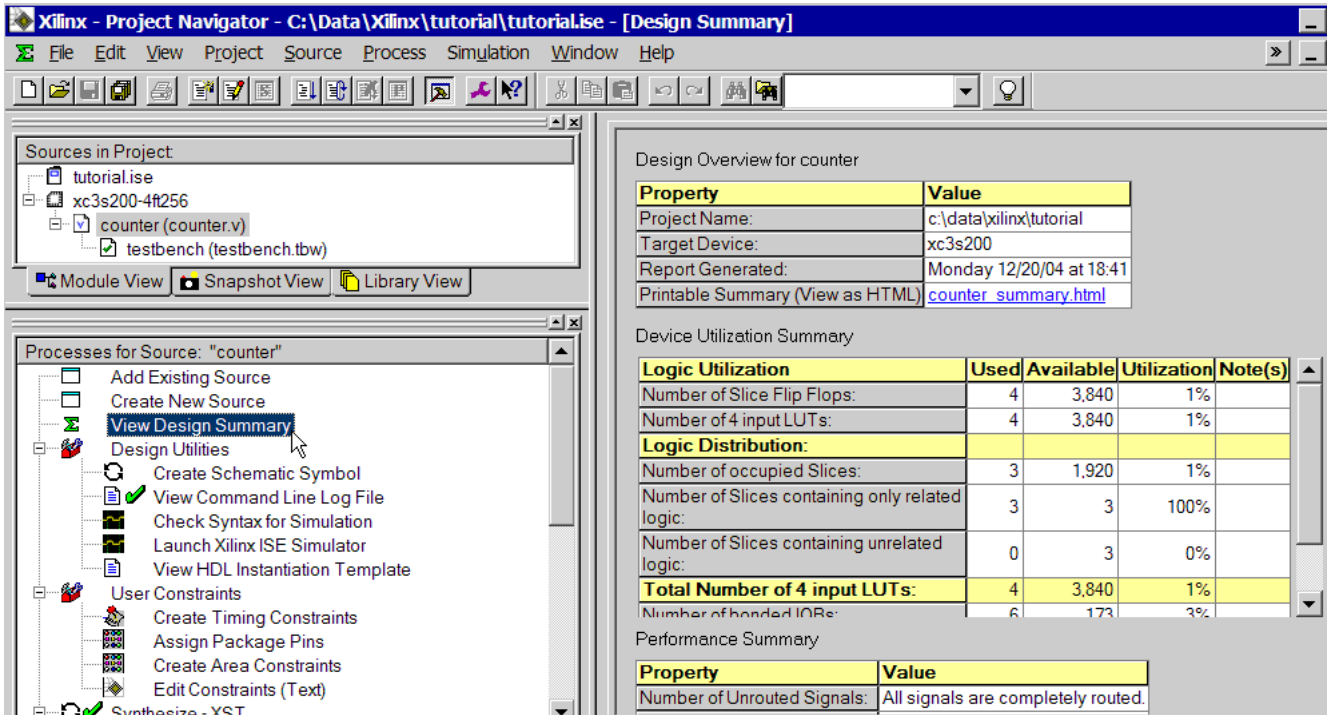

Figure 18: **Design Summary View**

2. In the Device Utilization Summary section, observe the number of Slice Flip Flops that were used during implementation. You should see 4 flip flops, since you implemented a 4-bit counter.
3. To see other reports, scroll to the bottom of the Design Summary. You can click on a report from here to view it in the ISE Text Editor.

Timing Closure

In this section, you will run timing analysis on your design to verify that your timing constraints were met. Timing closure is the process of working on your design to ensure that it meets your necessary timing requirements. ISE provides several tools to assist with timing closure.

1. In the Processes for Source window, under the Place & Route group of processes, expand the **Generate Post-Place & Route Static Timing** group by clicking the "**+**" sign.
2. Double-click the **Analyze Post-Place & Route Static Timing** process. The Timing Analyzer opens.
3. To analyze the design, select **Analyze** → **Against Timing Constraints...**. The Analyze with Timing Constraints dialog box opens.
4. Click **OK**.

When analysis is complete, the timing report opens.

5. Select **Timing summary** from the Timing Report Description tree in the left window. This displays the summary section of the timing report, where you can see that no timing errors were reported.

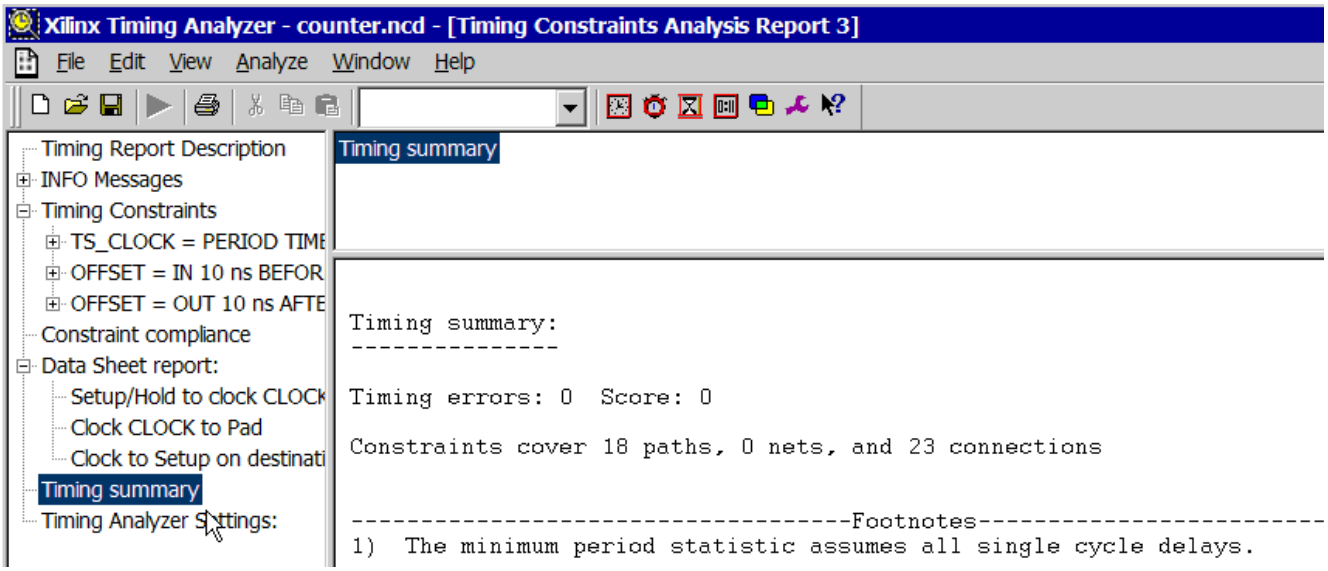

Figure 19: **Timing Analyzer - Timing Summary**

6. Close the Timing Analyzer without saving.

Viewing the Placed and Routed Design

In this section, you will use the FPGA Editor to view the design. You can view your design on the FPGA device, as well as edit the placement and routing with the FPGA Editor.

1. Double-click the **View/Edit Routed Design (FPGA Editor)** process found in the Place & Route group of processes. Your implemented design opens in the FPGA Editor.
2. Look in the **List** window to examine your design components.
3. Click on the **COUNT_OUT K12** IOB in the List window to select the row. This is one of the outputs in your design.

4. With the COUNT_OUT K12 row selected, select **View** → **Zoom Selection**. In the editor window, you can see the COUNT_OUT<0> IOB highlighted in red.
5. Push into (double-click) the red-highlighted **COUNT_OUT K12** IOB. You should see the following window:

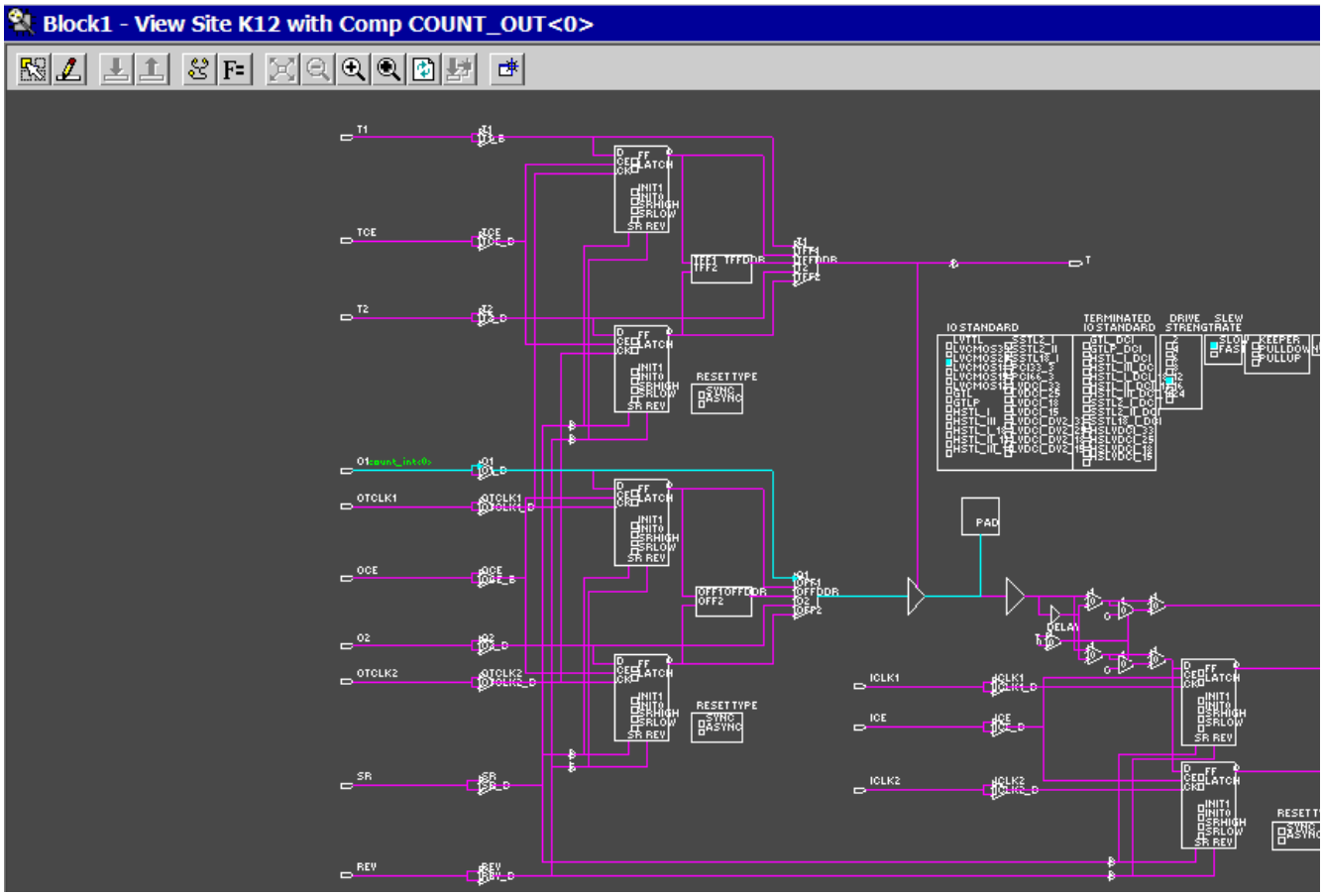

Figure 20: **FPGA Editor - Detailed View**

6. Enlarge the window and zoom in so you can see more detail.

This view shows the inside of an FPGA at the lowest viewable level. The blue line shows the route that is used through the IOB. The red lines show the routes that are available.

7. Verify that the signal goes to the pad as an output.
8. Close the FPGA Editor.

Timing Simulation (ISE Simulator)

You can verify that your design meets your timing requirements by running a timing simulation. You can use the same test bench waveform that was used earlier in the design flow for behavioral simulation.

When running timing simulation, the ISE tools create a structural HDL file which includes timing information available after Place and Route is run. The simulator will run on a model that is created based on the design to be downloaded to the FPGA.

If you are using ISE Base or Foundation, you can simulate your design with the ISE Simulator. To simulate your design with ModelSim, skip to the "Timing Simulation (ModelSim)" section.

To run the integrated simulation processes:

1. Select the **test bench** waveform in the Sources in Project window. You can see the ISE Simulator processes in the Processes for Source window.

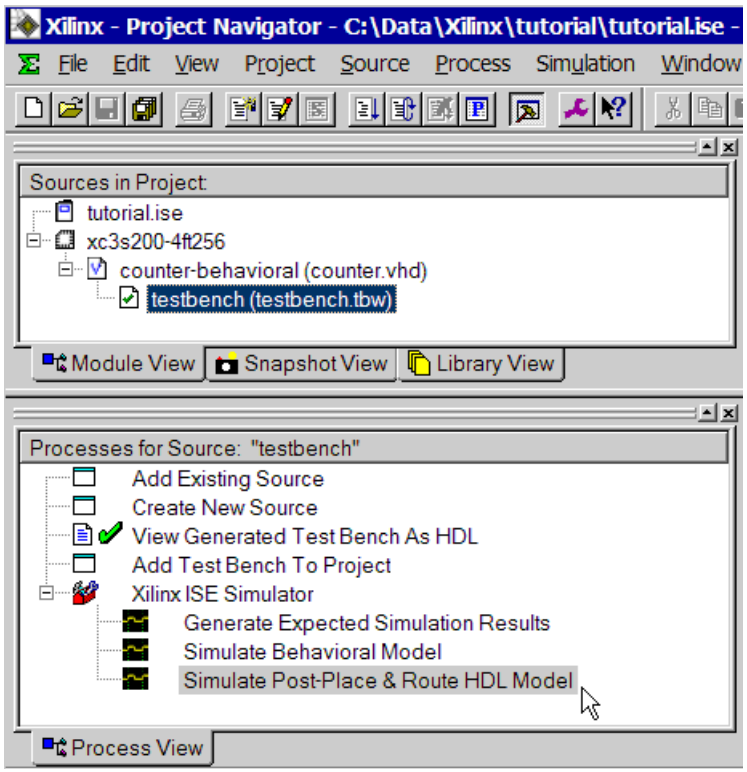

Figure 21: **Simulator Processes for Test Bench**

2. Double-click the **Simulate Post-Place & Route Model** process.

This process generates a timing-annotated netlist from the implemented design and simulates it. The resulting simulation is displayed in the Waveform Viewer. These results look different than those you saw in the behavioral simulation earlier in this tutorial. These results show timing delays.

The ISE window, including waveform view, should look like the following:

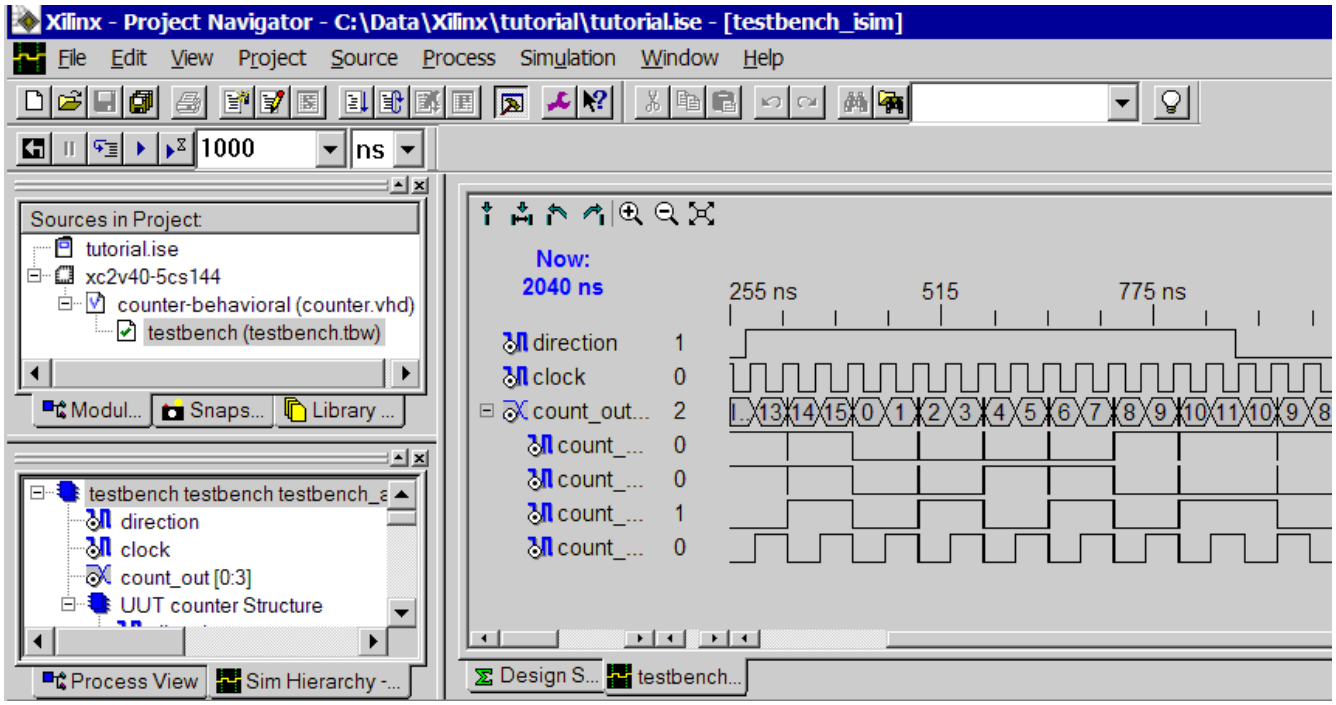

Figure 22: **Timing Simulation in ISE Simulator**

3. To see your simulation results, zoom in on the transitions and view the area between 300 ns and 900 ns to verify that the counter is counting up and down as directed by the stimulus on the DIRECTION port.
4. Zoom in again to see the timing delay between a rising clock edge and an output transition.

5. Click the **Measure Marker** button and then click near the 300 ns mark. Drag the second marker to the point where the output becomes stable to see the time delay between the clock edge and the transition.

Measure Marker

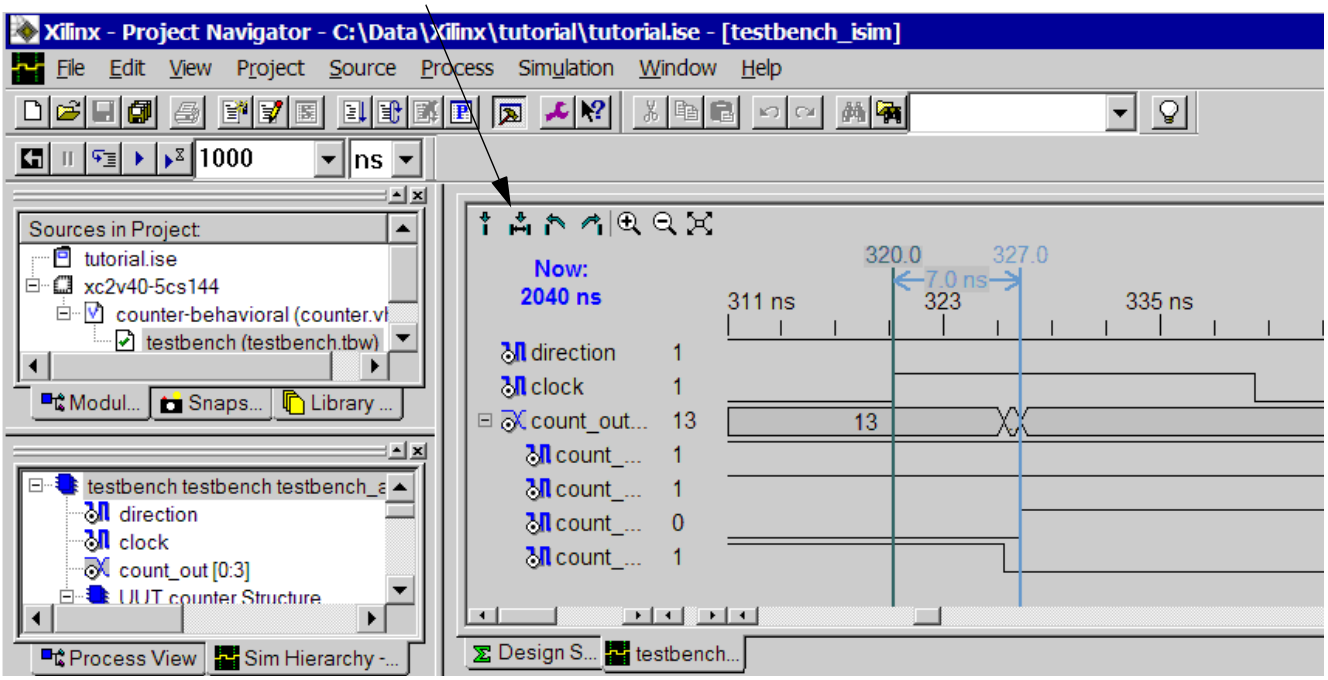

Figure 23: **View the Measure Marker**

6. Close the waveform view window.

You have completed timing simulation of your design using the ISE Simulator. Skip past the ModelSim section below, and proceed to the "Creating Configuration Data" section.

Timing Simulation (ModelSim)

If you have a ModelSim simulator installed, you can simulate your design using the integrated ModelSim flow. You can run processes from within ISE which launches the installed ModelSim simulator.

1. To run the integrated simulation processes, select the **test bench** in the Sources in Project window. You can see the ModelSim Simulator processes in the Processes for Source window.

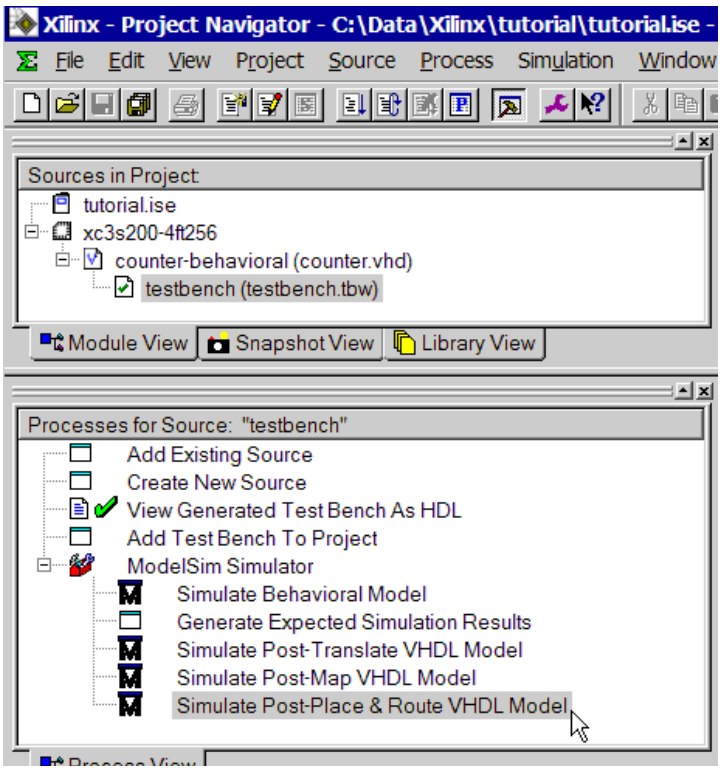

Figure 24: **Simulator Processes for Test Bench**

2. Double-click the **Simulate Post-Place & Route VHDL/Verilog Model** process.

Note: The implementation simulation processes will run the HDL language that you selected as your Generated Simulation Language when you created your project.

This process generates a timing-annotated netlist from the implemented design and simulates it in ModelSim. These results look different than those you saw in the behavioral simulation earlier in this tutorial. These results show timing delays.

The ModelSim window, including waveform, should look like the following:

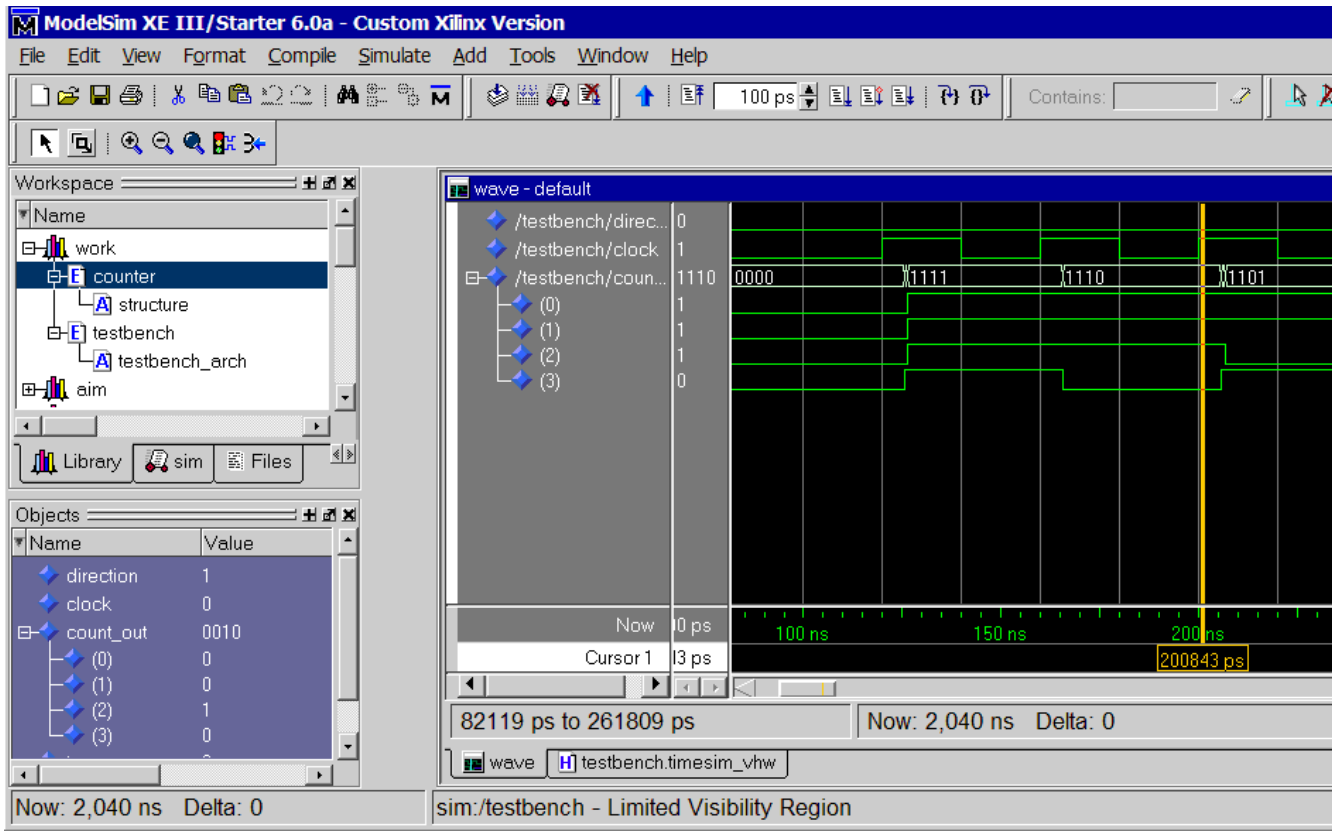

Figure 25: **Timing Simulation in ModelSim**

3. Zoom in on the area between 300 ns and 900 ns to verify that the counter is counting up and down as directed by the stimulus on the DIRECTION port.
4. Zoom in on the rising clock edges to see that the output transitions occur slightly later due to the timing delay.
5. Close the ModelSim window.

Creating Configuration Data

The final phase in the software flow is to generate a bitstream and configure the device.

Generating a Bitstream

The bitstream is a binary encoded file that is the equivalent of the design in a form that can be downloaded into the FPGA device.

1. Select the **counter** HDL source file in the Sources in Project window.
2. Run the **Generate Programming File** process located near the bottom of the Processes for Source window.

The bitstream is created by the Bitgen program. It is written into a file called `counter.bit`. This is the actual configuration data.

3. Select the **Design Summary** tab.

4. Scroll to the bottom of the Design Summary window and select **Bitgen Report** to view the report.

You are now ready to configure your device.

Configuring the Device

iMPACT is used to configure your FPGA or CPLD device. This is the last step in the design process. This section provides simple instructions for configuring a Spartan-3 xc3s200 device connected to your PC.

Note: Your board must be connected to your PC before proceeding. If the device on your board does not match the device assigned to the project, you will get errors. Please refer to the iMPACT Help for more information. To access the help, select **Help** → **Help Topics**.

To configure the device:

1. Click the "**+**" sign to expand the **Generate Programming File** processes.
2. Double-click the **Configure Device (iMPACT)** process.

iMPACT opens and the Configure Devices dialog box is displayed.

3. In the Configure Devices dialog box, verify that **Boundary-Scan Mode** is selected and click **Next**.
4. Verify that **Automatically connect to cable and identify Boundary-Scan chain** is selected and click **Finish**.
5. If you get a message saying that there are two devices found, click **OK** to continue.
6. The **Assign New Configuration File** dialog box appears. Assign a configuration file to each device in the JTAG chain. Select the `counter.bit` file and click **Open**. If you want to skip any devices in this process, click **Bypass** to continue.
7. If you get a Warning message, click **OK**.
8. Right-click on the counter device image, and select **Program...** to open the **Program Options** dialog box.
9. Click **OK** to program the device.

ISE programs the device and displays Programming Succeeded if the operation was successful. On the device, four outputs should be active.

10. Close iMPACT without saving.

You have completed the ISE Quick Start Tutorial. For an in-depth explanation of the ISE design tools, see the ISE In-Depth Tutorial on the Xilinx® web site at:
http://www.xilinx.com/support/techsup/tutorials/