

ISE 8.2i Quick Start Tutorial



Xilinx is disclosing this Document and Intellectual Property (hereinafter “the Design”) to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED “AS IS” WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

Copyright © 1995-2006 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM, Inc. All other trademarks are the property of their respective owners.

About This Tutorial

The ISE 8.2i 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 8.2i software.

Additional Resources

To find additional documentation, see the Xilinx website at:

<http://www.xilinx.com/literature>.

To search the Answer Database of silicon, software, and IP questions and answers, or to create a technical support WebCase, see the Xilinx website at:

<http://www.xilinx.com/support>.

Table of Contents

Preface: About This Tutorial

Additional Resources	3
----------------------------	---

ISE 8.2i Quick Start Tutorial

Getting Started	7
Software Requirements	7
Hardware Requirements	7
Starting the ISE Software	8
Accessing Help	8
Create a New Project	9
Create an HDL Source	10
Creating a VHDL Source	10
Using Language Templates (VHDL)	11
Final Editing of the VHDL Source	12
Creating a Verilog Source	13
Using Language Templates (Verilog)	14
Final Editing of the Verilog Source	15
Checking the Syntax of the New Counter Module	15
Design Simulation	16
Verifying Functionality using Behavioral Simulation	16
Create a Self-Checking Test Bench Waveform	18
Simulating Design Functionality	19
Create Timing Constraints	20
Entering Timing Constraints	21
Implement Design and Verify Constraints	24
Implementing the Design	24
Assigning Pin Location Constraints	25
Reimplement Design and Verify Pin Locations	26
Verify Design using Timing Simulation	27
Download Design to the Spartan™-3 Demo Board	28

ISE 8.2i Quick Start Tutorial

The ISE 8.2i Quick Start Tutorial provides Xilinx PLD designers with a quick overview of the basic design process using ISE 8.2i. After you have completed the tutorial, you will have an understanding of how to create, verify, and implement a design.

Note: This tutorial is designed for ISE 8.2i on Windows.

This tutorial contains the following sections:

- “Getting Started”
- “Create a New Project”
- “Create an HDL Source”
- “Design Simulation”
- “Create Timing Constraints”
- “Implement Design and Verify Constraints”
- “Reimplement Design and Verify Pin Locations”
- “Download Design to the Spartan™-3 Demo Board”

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/>

Getting Started

Software Requirements

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

- ISE 8.2i

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

Hardware Requirements

To use this tutorial, you must have the following hardware:

- Spartan-3 Startup Kit, containing the Spartan-3 Startup Kit Demo Board

Starting the ISE Software

To start ISE, double-click the desktop icon,



or start ISE from the Start menu by selecting:

Start → All Programs → Xilinx ISE 8.2i → Project Navigator

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

Accessing Help

At any time during the tutorial, you can access online help for additional information about the ISE software and related tools.

To open Help, 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.

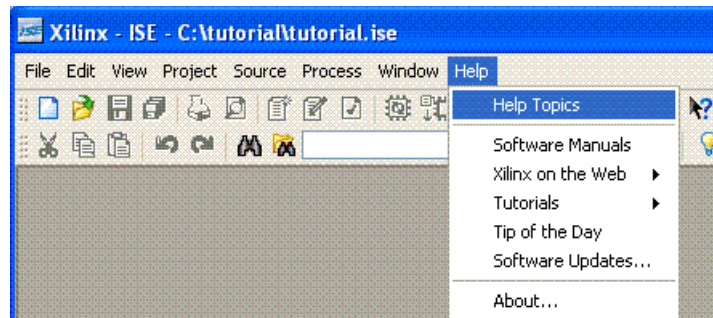


Figure 1: ISE Help Topics

Create a New Project

Create a new ISE project which will target the FPGA device on the Spartan-3 Startup Kit demo board.

To create a new project:

1. Select **File > New Project...** The New Project Wizard appears.
2. Type **tutorial** in the Project Name field.
3. Enter or browse to a location (directory path) for the new project. A tutorial subdirectory is created automatically.
4. Verify that **HDL** is selected from the Top-Level Source Type list.
5. Click **Next** to move to the device properties page.
6. Fill in the properties in the table as shown below:
 - ◆ Product Category: **All**
 - ◆ Family: **Spartan3**
 - ◆ Device: **XC3S200**
 - ◆ Package: **FT256**
 - ◆ Speed Grade: **-4**
 - ◆ Top-Level Module Type: **HDL**
 - ◆ Synthesis Tool: **XST (VHDL/Verilog)**
 - ◆ Simulator: **ISE Simulator (VHDL/Verilog)**
 - ◆ Verify that **Enable Enhanced Design Summary** is selected.

Leave the default values in the remaining fields.

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

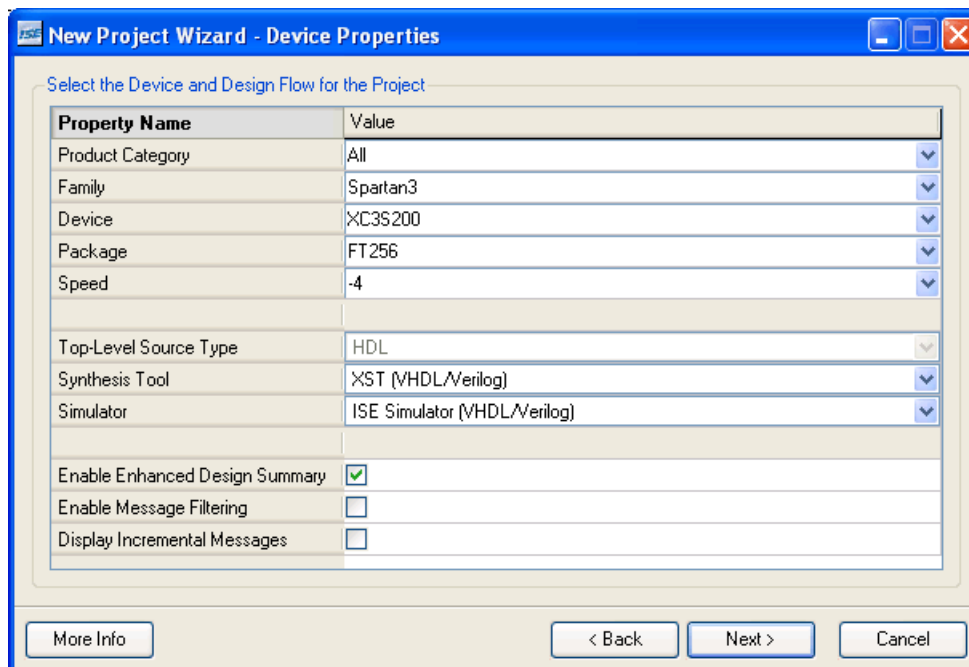


Figure 2: Project Device Properties

- 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 complete.

Create an HDL Source

In this section, you will create the 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.

Creating a VHDL Source

Create a VHDL source file for the project as follows:

- Click the **New Source** button in the New Project Wizard.
- Select **VHDL Module** as the source type.
- Type in the file name **counter**.
- Verify that the **Add to project** checkbox is selected.
- Click **Next**.
- Declare the ports for the counter design by filling in the port information as shown below:

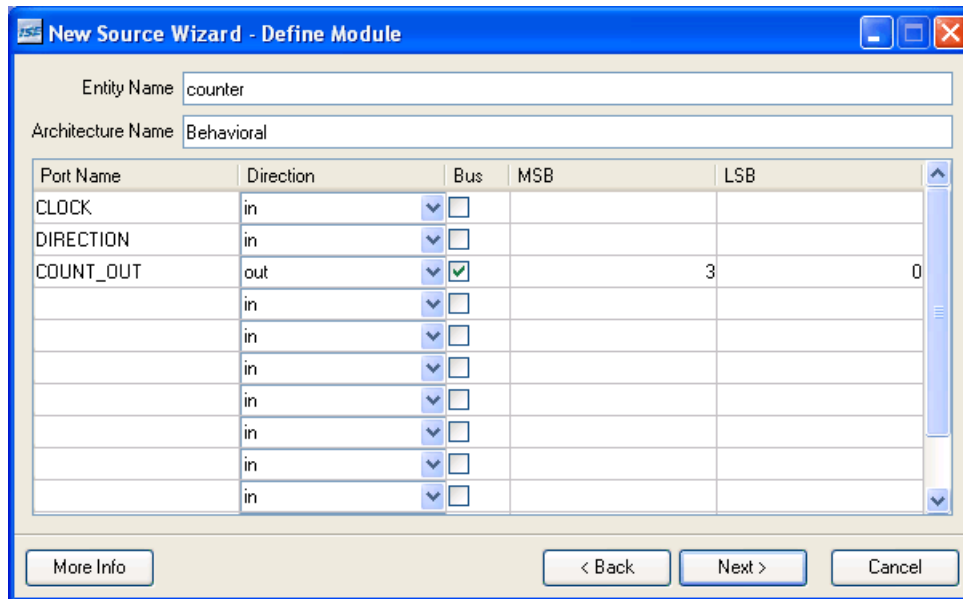


Figure 3: Define Module

- Click **Next**, then **Finish** in the New Source Information dialog box to complete the new source file template.
- Click **Next**, then **Next**, then **Finish**.

The source file containing the entity/architecture pair displays in the Workspace, and the counter displays in the Source tab, as shown below:

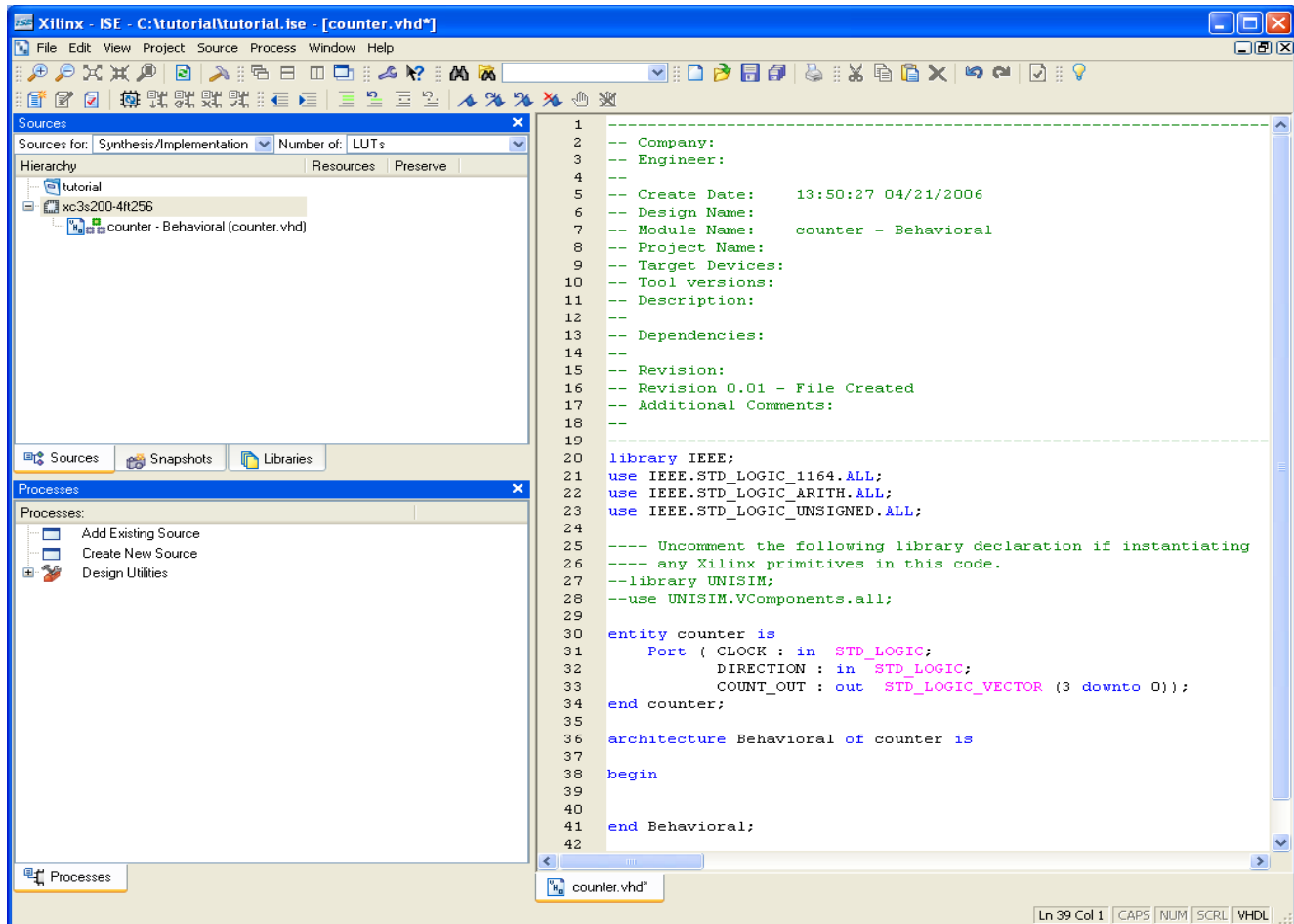


Figure 4: New Project in ISE

Using Language Templates (VHDL)

The next step in creating the new source is to add the behavioral description for the counter. To do this you will use a simple counter code example from the ISE Language Templates and customize it for the counter design.

1. Place the cursor just below the `begin` statement within the counter `architecture`.
2. Open the Language Templates by selecting **Edit** → **Language Templates...**
Note: You can tile the Language Templates and the counter file by selecting **Window** → **Tile Vertically** to make them both visible.
3. Using the “+” symbol, browse to the following code example:
VHDL → **Synthesis Constructs** → **Coding Examples** → **Counters** → **Binary** → **Up/Down Counters** → **Simple Counter**
4. With Simple Counter selected, select **Edit** → **Use in File**, or select the **Use Template in File** toolbar button. This step copies the template into the counter source file.



5. Close the Language Templates.

Final Editing of the VHDL Source

1. Add the following signal declaration to handle the feedback of the counter output below the `architecture` declaration and above the first `begin` statement:

```
signal count_int : std_logic_vector(3 downto 0) := "0000";
```
2. Customize the source file for the counter design by replacing the port and signal name placeholders with the actual ones as follows:
 - ◆ replace all occurrences of `<clock>` with `CLOCK`
 - ◆ replace all occurrences of `<count_direction>` with `DIRECTION`
 - ◆ replace all occurrences of `<count>` with `count_int`
3. Add the following line below the `end process;` statement:

```
COUNT_OUT <= count_int;
```
4. Save the file by selecting **File** → **Save**.

When you are finished, the counter source file will look like the following:

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

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitive in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity counter is
  Port ( CLOCK : in  STD_LOGIC;
        DIRECTION : in  STD_LOGIC;
        COUNT_OUT : out  STD_LOGIC_VECTOR (3 downto 0));
end counter;

architecture Behavioral of counter is
  signal count_int : std_logic_vector(3 downto 0) := "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

Create the top-level Verilog source file for the project as follows:

1. Click **New Source** in the New Project dialog box.
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. Declare the ports for the counter design by filling in the port information as shown below:

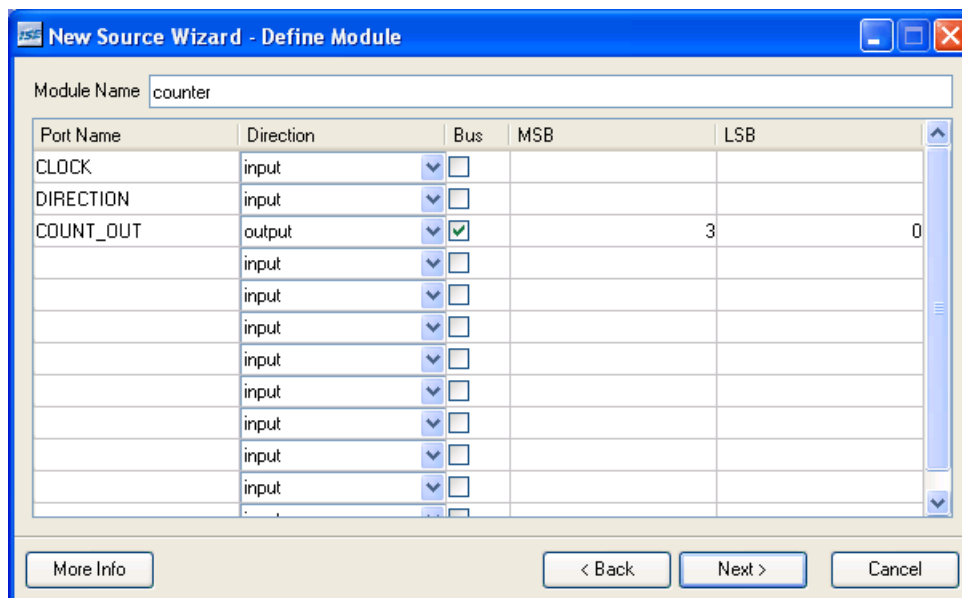


Figure 5: Define Module

7. Click **Next**, then **Finish** in the New Source Information dialog box to complete the new source file template.
8. Click **Next**, then **Next**, then **Finish**.

The source file containing the counter module displays in the Workspace, and the counter displays in the Sources tab, as shown below:

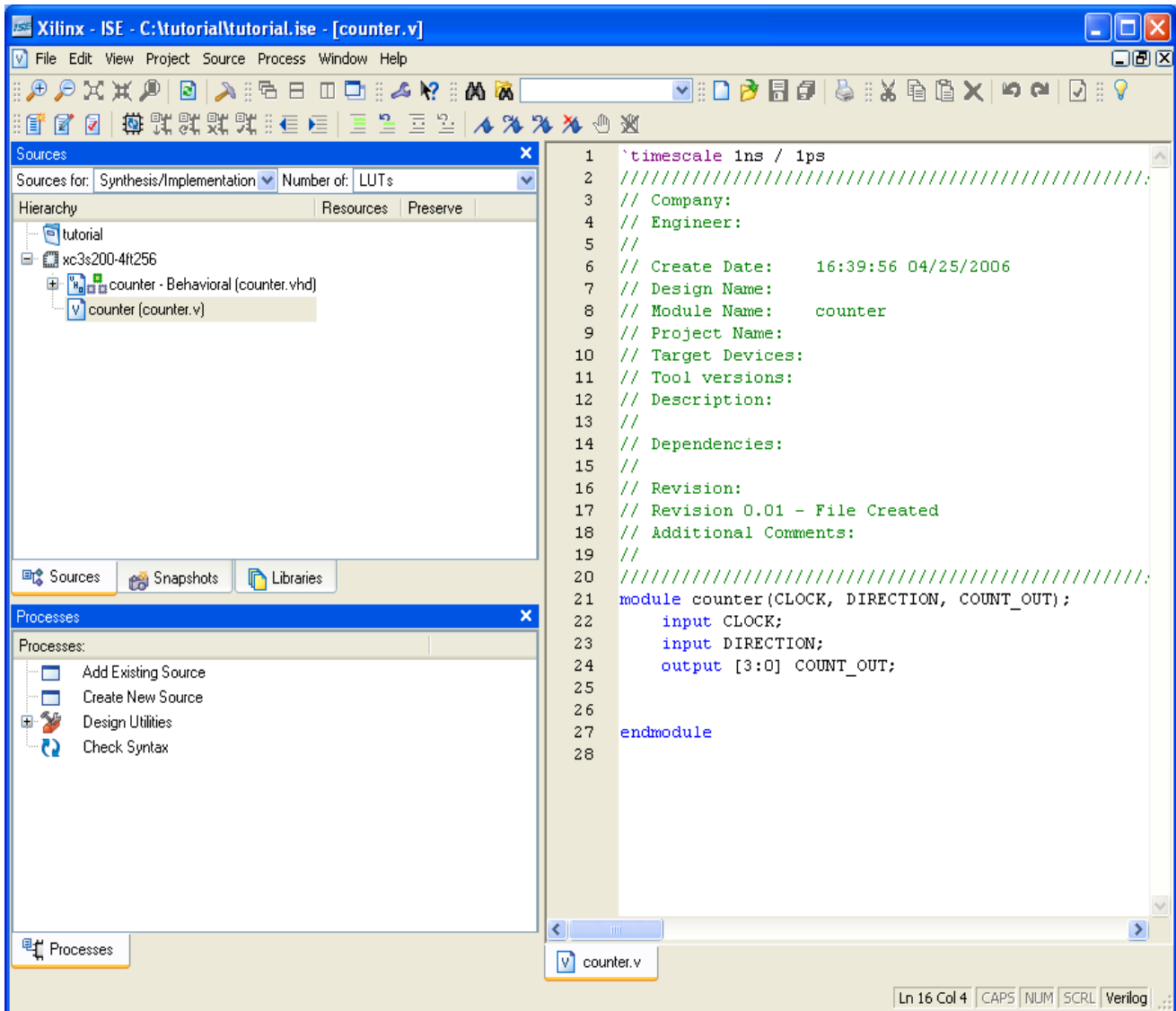


Figure 6: New Project in ISE

Using Language Templates (Verilog)

The next step in creating the new source is to add the behavioral description for counter. Use a simple counter code example from the ISE Language Templates and customize it for the counter design.

1. Place the cursor on the line below the `output [3:0] COUNT_OUT;` statement.
2. Open the Language Templates by selecting **Edit** → **Language Templates...**

Note: You can tile the Language Templates and the counter file by selecting **Window** → **Tile Vertically** to make them both visible.

3. Using the “+” symbol, browse to the following code example:

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

4. With Simple Counter selected, select **Edit → Use in File**, or select the **Use Template in File** toolbar button. This step copies the template into the counter source file.



5. Close the Language Templates.

Final Editing of the Verilog Source

1. To declare and initialize the register that stores the counter value, modify the declaration statement in the first line of the template as follows:
replace: `reg [<upper>:0] <reg_name>;`
with: `reg [3:0] count_int = 0;`
2. Customize the template for the counter design by replacing the port and signal name placeholders with the actual ones as follows:
 - ◆ replace all occurrences of `<clock>` with `CLOCK`
 - ◆ replace all occurrences of `<up_down>` with `DIRECTION`
 - ◆ replace all occurrences of `<reg_name>` with `count_int`
3. Add the following line just above the `endmodule` statement to assign the register value to the output port:
`assign COUNT_OUT = count_int;`
4. Save the file by selecting **File → Save**.

When you are finished, the code for the counter will 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, check the syntax of the design to find errors and typos.

1. Verify that **Synthesis/Implementation** is selected from the drop-down list in the Sources window.

2. Select the **counter** design source in the Sources window to display the related processes in the Processes window.
3. Click the “+” next to the Synthesize-XST process to expand the process group.
4. Double-click the **Check Syntax** process.
Note: You must correct any errors found in your source files. You can check for errors in the Console tab of the Transcript window. If you continue without valid syntax, you will not be able to simulate or synthesize your design.
5. Close the HDL file.

Design Simulation

Verifying Functionality using Behavioral Simulation

Create a test bench waveform containing input stimulus you can use to verify the functionality of the counter module. The test bench waveform is a graphical view of a test bench.

Create the test bench waveform as follows:

1. Select the **counter** HDL file in the Sources window.
2. Create a new test bench source by selecting **Project** → **New Source**.
3. In the New Source Wizard, select **Test Bench WaveForm** as the source type, and type **counter_tbw** in the File Name field.
4. Click **Next**.
5. The Associated Source page shows that you are associating the test bench waveform with the source file `counter`. Click **Next**.
6. The Summary page shows that the source will be added to the project, and it displays the source directory, type and name. Click **Finish**.
7. You need to set the clock frequency, setup time and output delay times in the Initialize Timing dialog box before the test bench waveform editing window opens.

The requirements for this design are the following:

- ◆ The counter must operate correctly with an input clock frequency = 25 MHz.
- ◆ The DIRECTION input will be valid 10 ns before the rising edge of CLOCK.
- ◆ The output (COUNT_OUT) must be valid 10 ns after the rising edge of CLOCK.

The design requirements correspond with the values below.

Fill in the fields in the Initialize Timing dialog box with the following information:

- ◆ Clock Time High: **20** ns.
- ◆ Clock Time Low: **20** ns.
- ◆ Input Setup Time: **10** ns.
- ◆ Output Valid Delay: **10** ns.
- ◆ Offset: **0** ns.
- ◆ Global Signals: **GSR (FPGA)**

Note: When GSR(FPGA) is enabled, 100 ns. is added to the Offset value automatically.

- ◆ Initial Length of Test Bench: **1500** ns.

Leave the default values in the remaining fields.

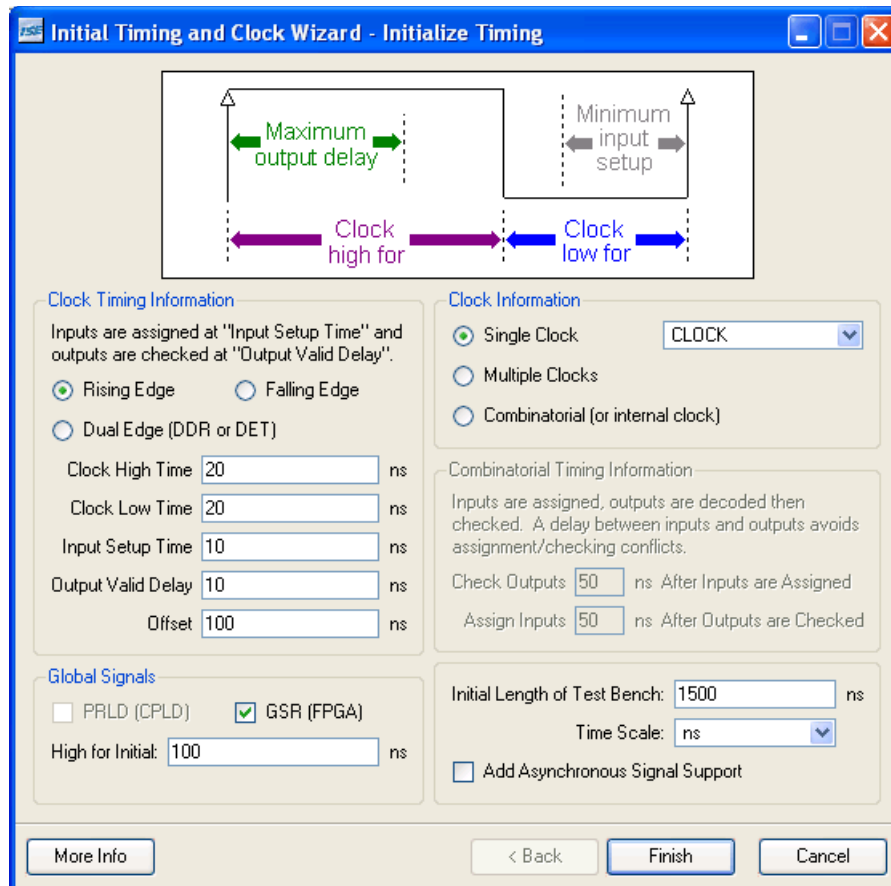


Figure 7: Initialize Timing

8. Click **Finish** to complete the timing initialization.
9. The blue shaded areas that precede the rising edge of the CLOCK correspond to the Input Setup Time in the Initialize Timing dialog box. Toggle the DIRECTION port to define the input stimulus for the counter design as follows:
 - ◆ Click on the blue cell at approximately the **300** ns to assert DIRECTION high so that the counter will count up.
 - ◆ Click on the blue cell at approximately the **900** ns to assert DIRECTION high so that the counter will count down.

Note: For more accurate alignment, you can use the **Zoom In** and **Zoom Out** toolbar buttons.

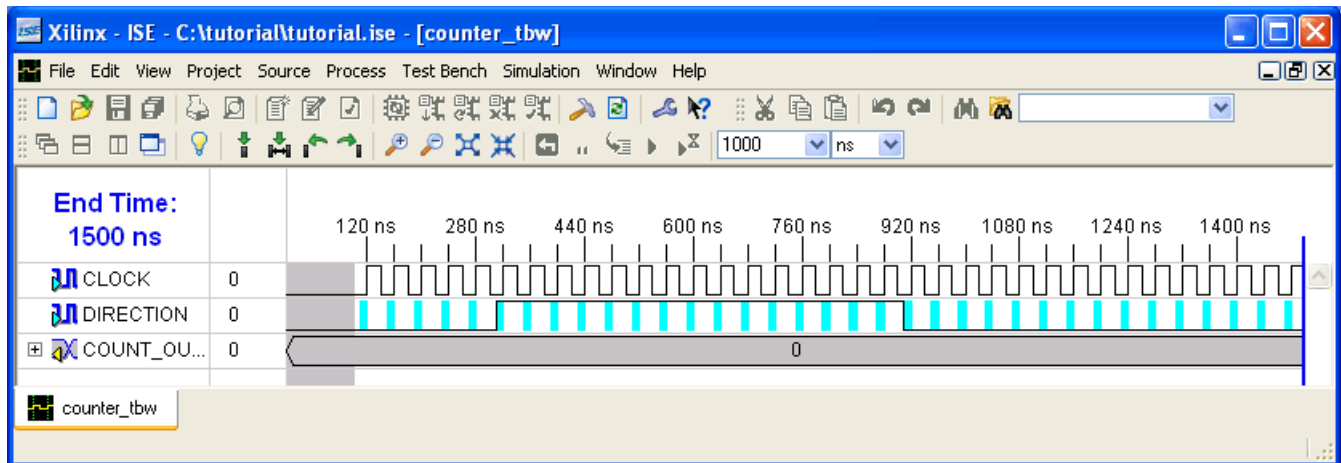


Figure 8: Test Bench Waveform

10. Save the waveform.
11. In the Sources window, select the **Behavioral Simulation** view to see that the test bench waveform file is automatically added to your project.

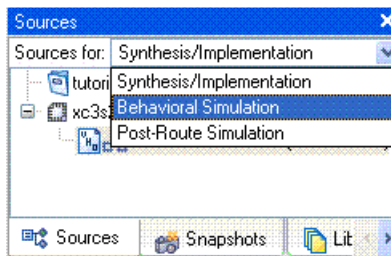


Figure 9: Behavior Simulation Selection

12. Close the test bench waveform.

Create a Self-Checking Test Bench Waveform

Add the expected output values to finish creating the test bench waveform. This transforms the test bench waveform into a self-checking test bench waveform. The key benefit to a self-checking test bench waveform is that it compares the desired and actual output values and flags errors in your design as it goes through the various transformations, from behavioral HDL to the device specific representation.

To create a self-checking test bench, edit output values manually, or run the Generate Expected Results process to create them automatically. If you run the Generate Expected Results process, visually inspect the output values to see if they are the ones you expected for the given set of input values.

To create the self-checking test bench waveform automatically, do the following:

1. Verify that **Behavioral Simulation** is selected from the drop-down list in the Sources window.
2. Select the **counter_tbw** file in the Sources window.

- In the Processes tab, click the “+” to expand the Xilinx ISE Simulator process and double-click the **Generate Expected Simulation Results** process. This process simulates the design in a background process.
- The **Expected Results** dialog box opens. Select **Yes** to annotate the results to the test bench.

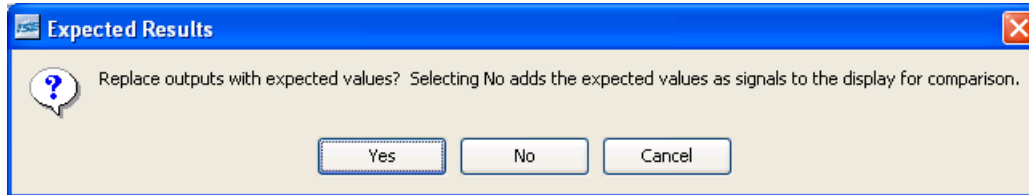


Figure 10: Expected Results Dialog Box

- Click the “+” to expand the COUNT_OUT bus and view the transitions that correspond to the Output Delay value (yellow cells) specified in the Initialize Timing dialog box.

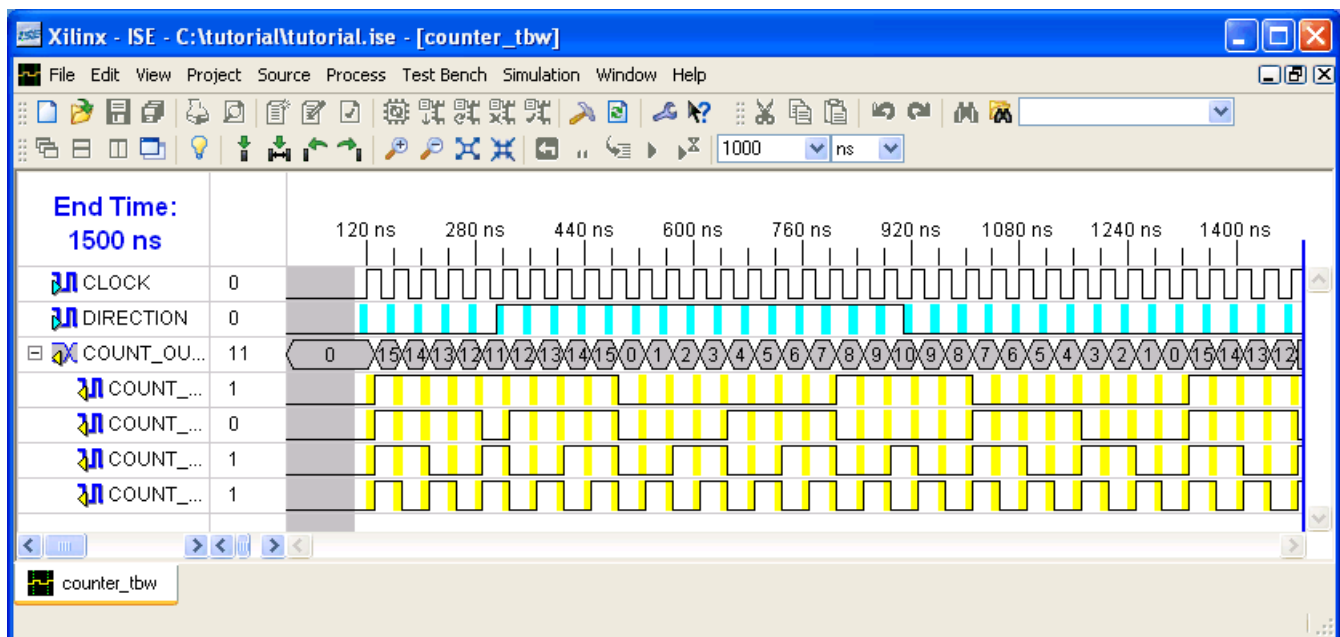


Figure 11: Test Bench Waveform with Results

- Save the test bench waveform and close it.
You have now created a self-checking test bench waveform.

Simulating Design Functionality

Verify that the counter design functions as you expect by performing behavior simulation as follows:

- Verify that **Behavioral Simulation** and **counter_tbw** are selected in the Sources window.
- In the **Processes** tab, click the “+” to expand the Xilinx ISE Simulator process and double-click the **Simulate Behavioral Model** process.

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

- To view your simulation results, select the **Simulation** tab and zoom in on the transitions.

The simulation waveform results will look like the following:

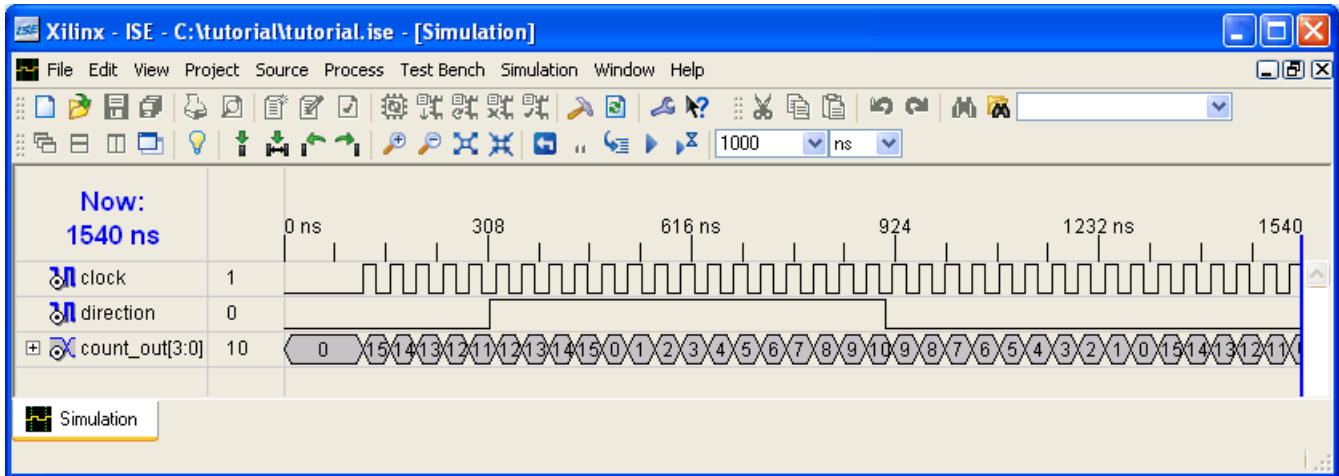


Figure 12: Simulation Results

Note: You can ignore any rows that start with **TX**.

- Verify that the counter is counting up and down as expected.
- Close the simulation view. If you are prompted with the following message, "You have an active simulation open. Are you sure you want to close it?", click **Yes** to continue.

You have now completed simulation of your design using the ISE Simulator.

Create Timing Constraints

Specify the timing between the FPGA and its surrounding logic as well as the frequency the design must operate at internal to the FPGA. The timing is specified by entering constraints that guide the placement and routing of the design. It is recommended that you enter global constraints. The clock period constraint specifies the clock frequency at which your design must operate inside the FPGA. The offset constraints specify when to expect valid data at the FPGA inputs and when valid data will be available at the FPGA outputs.

Entering Timing Constraints

To constrain the design do the following:

1. Select **Synthesis/Implementation** from the drop-down list in the Sources window.
2. Select the **counter** HDL source file.
3. Click the “+” sign next to the User Constraints processes group, and 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:

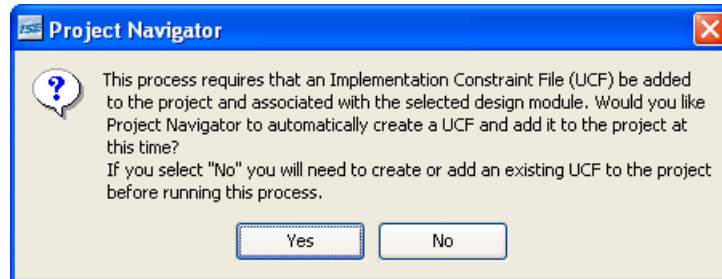


Figure 13: Prompt to Add UCF File to Project

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 window. The Xilinx Constraints Editor opens automatically.

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

In the next step, enter values in the fields associated with **CLOCK** in the Constraints Editor Global tab.

5. Select **CLOCK** in the Clock Net Name field, then select the **Period** toolbar button or double-click the empty Period field to display the Clock Period dialog box.



- Enter **40 ns** in the Time field.

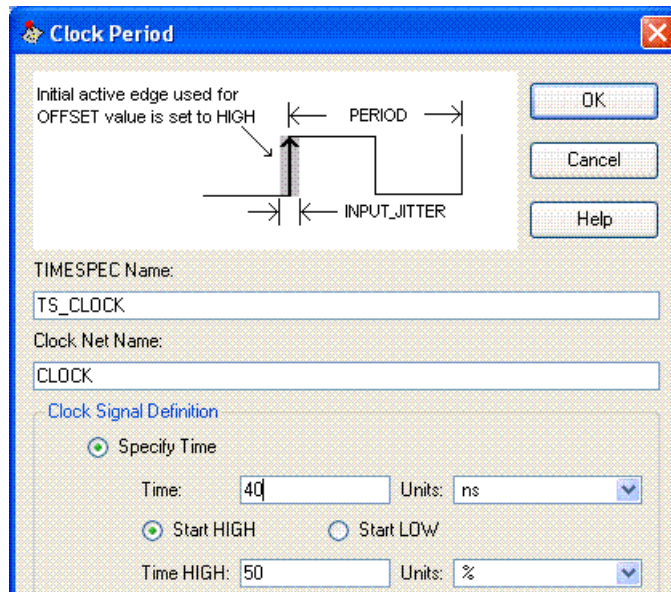


Figure 14: Clock Period

- Click **OK**.
- Select the **Pad to Setup** toolbar button or double-click the empty **Pad to Setup** field to display the Pad to Setup dialog box.



- Enter **10 ns** in the OFFSET field to set the input offset constraint.

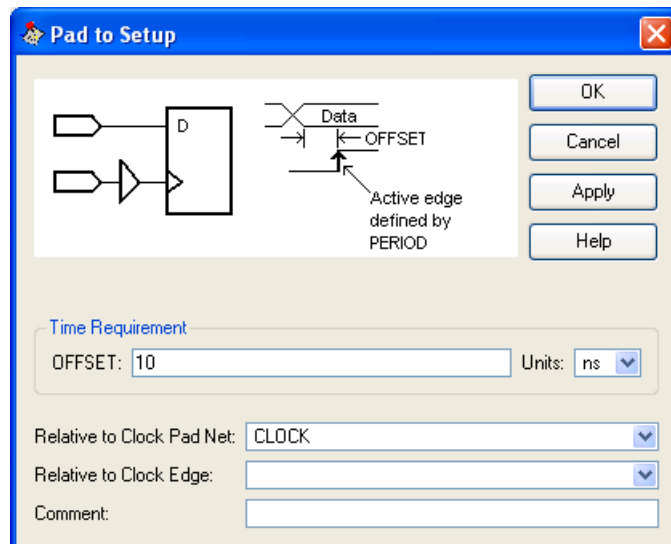


Figure 15: Pad to Setup

- Click **OK**.

11. Select the **Clock to Pad** toolbar button or double-click the empty **Clock to Pad** field to display the Clock to Pad dialog box.



12. Enter **10 ns** in the OFFSET field to set the output delay constraint.

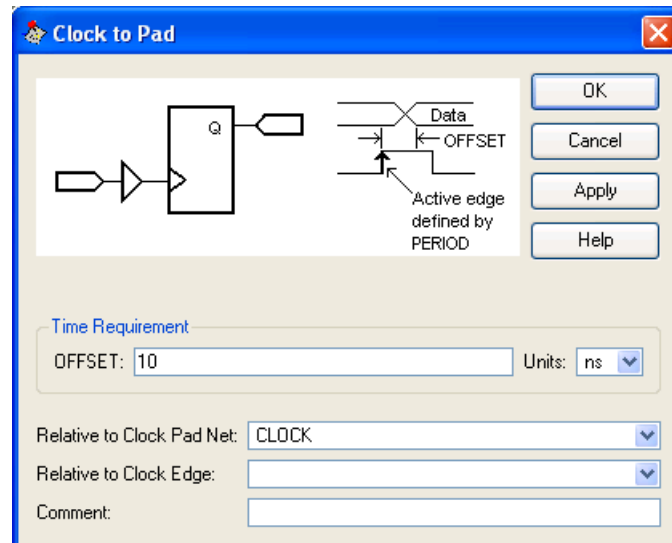


Figure 16: **Clock to Pad**

13. Click **OK**.

The constraints are displayed in the Constraints (read-write) tab, as shown below:

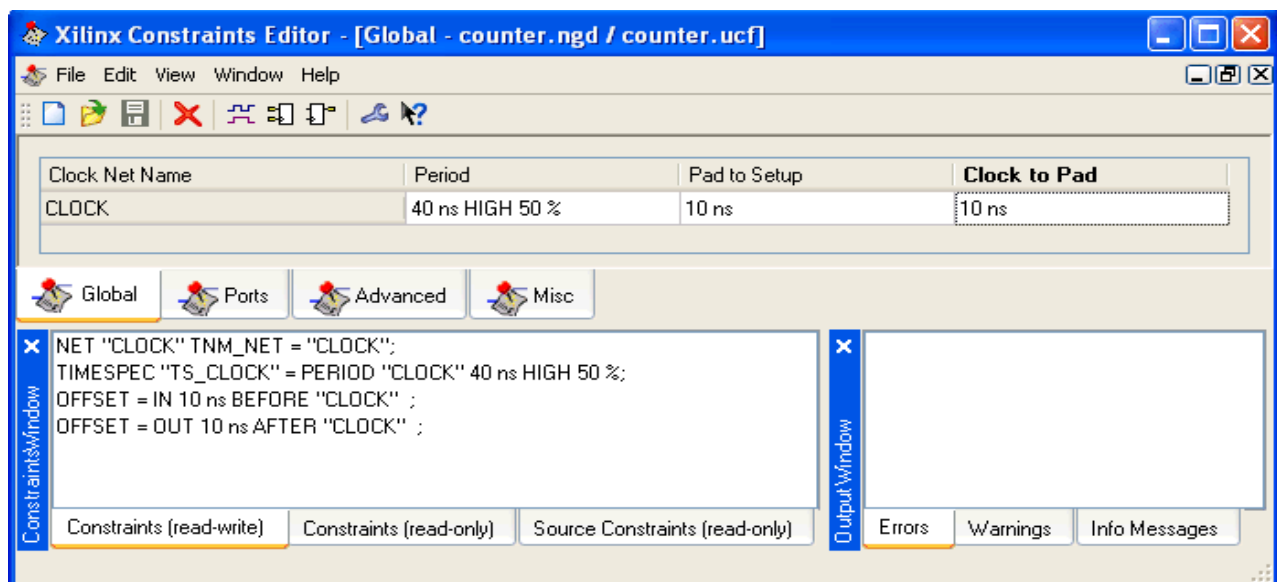


Figure 17: **Timing Constraints**

14. Save the timing constraints. If you are prompted to rerun the TRANSLATE or XST step, click **OK** to continue.
15. Close the Constraints Editor.

Implement Design and Verify Constraints

Implement the design and verify that it meets the timing constraints specified in the previous section.

Implementing the Design

1. Select the **counter** source file in the Sources window.
2. Open the Design Summary by double-clicking the **View Design Summary** process in the Processes tab.
3. Double-click the **Implement Design** process in the Processes tab.
4. Notice that after Implementation is complete, the Implementation processes have a green check mark next to them indicating that they completed successfully without Errors or Warnings.

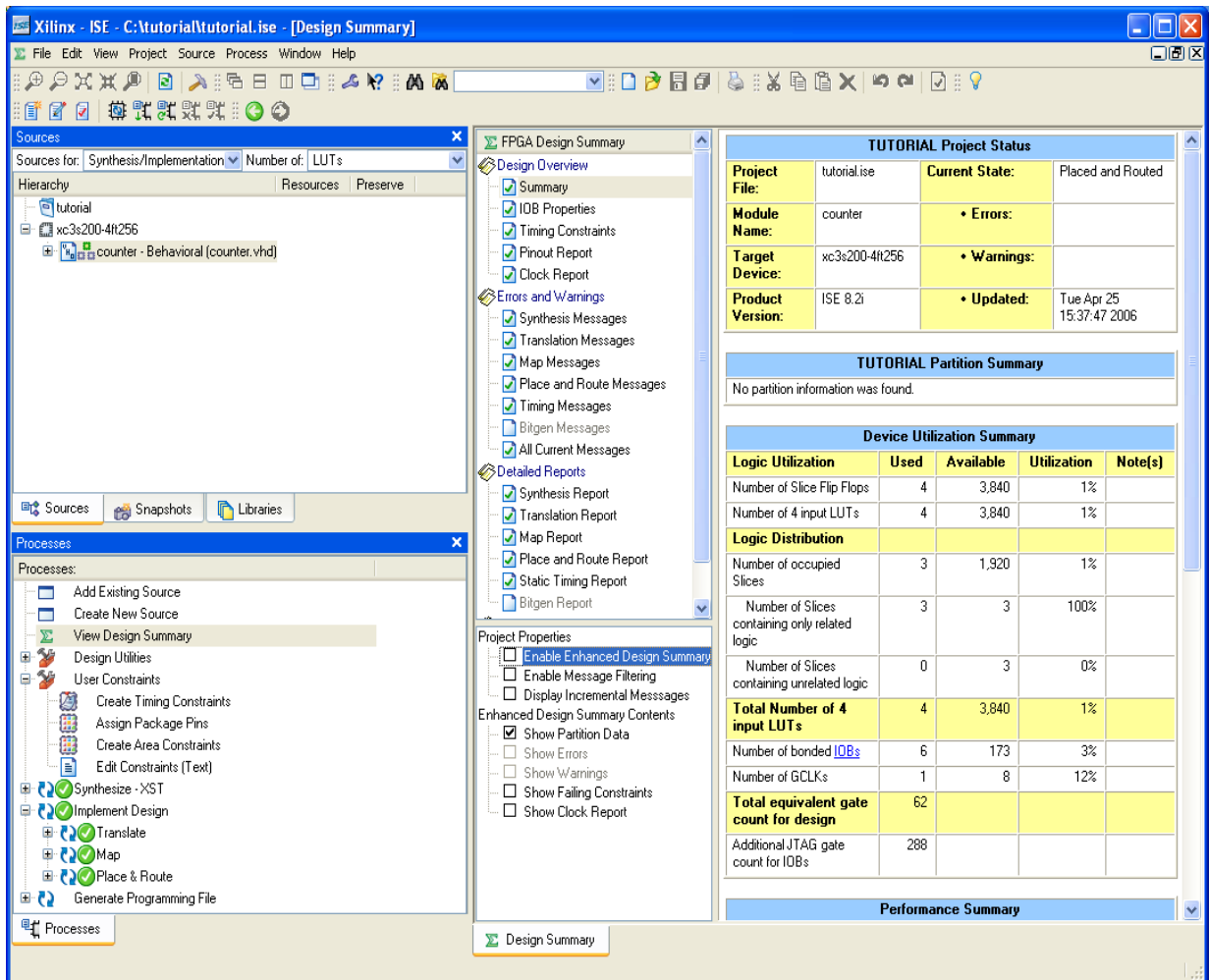
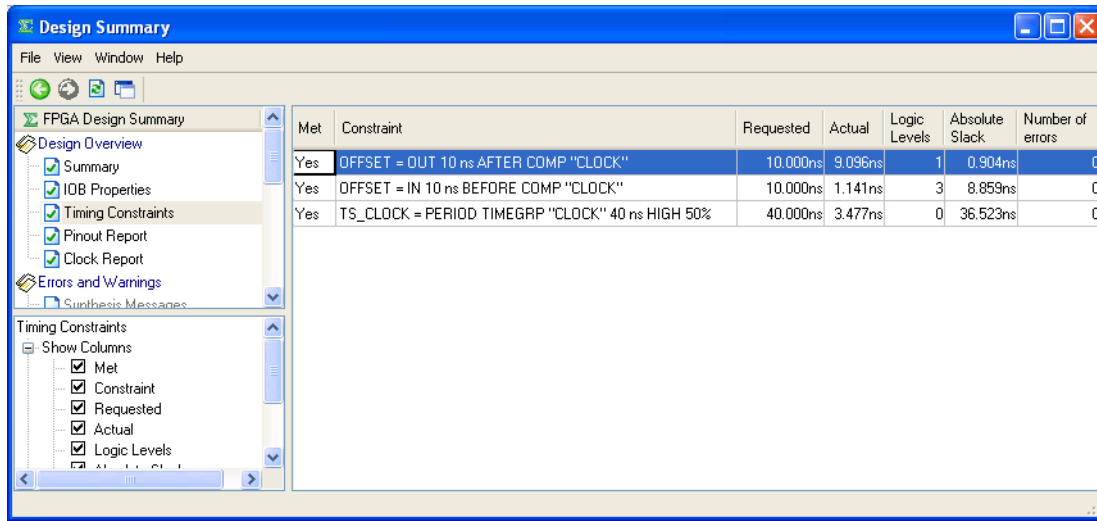


Figure 18: Post Implementation Design Summary

5. Locate the **Performance Summary** table near the bottom of the Design Summary.

- Click the **All Constraints Met** link in the Timing Constraints field to view the Timing Constraints report. Verify that the design meets the specified timing requirements.



Met	Constraint	Requested	Actual	Logic Levels	Absolute Slack	Number of errors
Yes	OFFSET = OUT 10 ns AFTER COMP "CLOCK"	10.000ns	9.096ns	1	0.904ns	0
Yes	OFFSET = IN 10 ns BEFORE COMP "CLOCK"	10.000ns	1.141ns	3	8.859ns	0
Yes	TS_CLOCK = PERIOD TIMEGRP "CLOCK" 40 ns HIGH 50%	40.000ns	3.477ns	0	36.523ns	0

Figure 19: All Constraints Met Report

- Close the Design Summary.

Assigning Pin Location Constraints

Specify the pin locations for the ports of the design so that they are connected correctly on the Spartan-3 Startup Kit demo board.

To constrain the design ports to package pins, do the following:

- Verify that **counter** is selected in the Sources window.
- Double-click the **Assign Package Pins** process found in the User Constraints process group. The Xilinx Pinout and Area Constraints Editor (PACE) opens.
- Select the **Package View** tab.
- In the Design Object List window, enter a pin location for each pin in the **Loc** column using the following information:
 - CLOCK input port connects to FPGA pin **T9** (GCK0 signal on board)
 - COUNT_OUT<0> output port connects to FPGA pin **K12** (LD0 signal on board)
 - COUNT_OUT<1> output port connects to FPGA pin **P14** (LD1 signal on board)
 - COUNT_OUT<2> output port connects to FPGA pin **L12** (LD2 signal on board)
 - COUNT_OUT<3> output port connects to FPGA pin **N14** (LD3 signal on board)
 - DIRECTION input port connects to FPGA pin **K13** (SW7 signal on board)

Notice that the assigned pin locations are shown in blue:

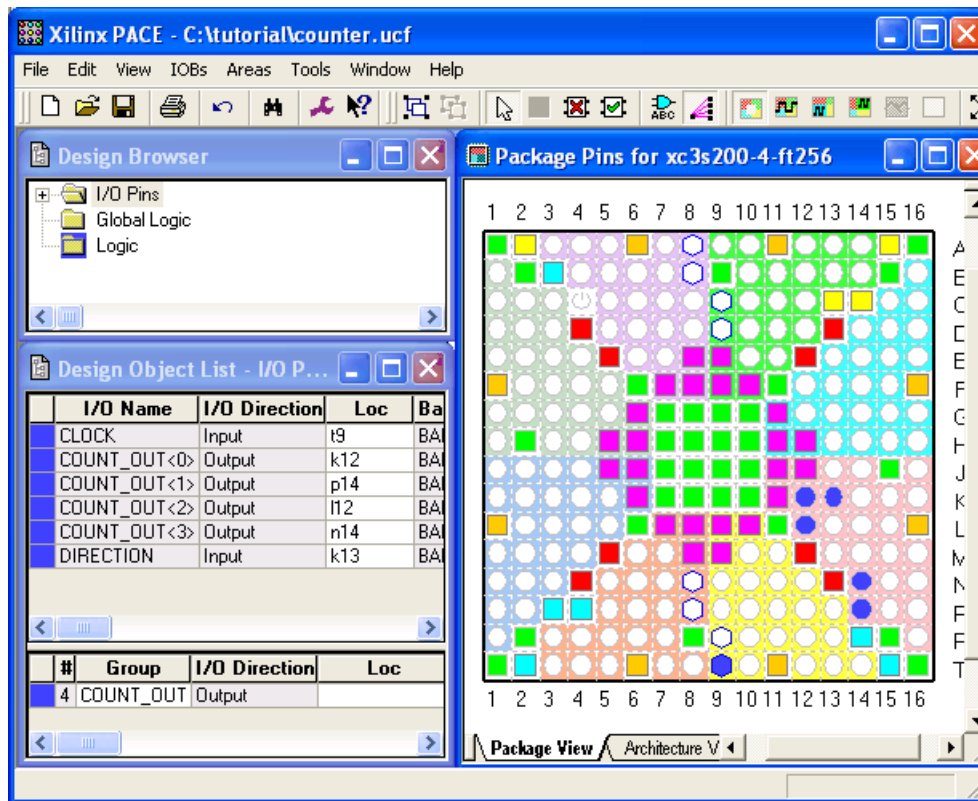


Figure 20: Package Pin Locations

5. 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**.
6. Close PACE.

Notice that the Implement Design processes have an orange question mark next to them, indicating they are out-of-date with one or more of the design files. This is because the UCF file has been modified.

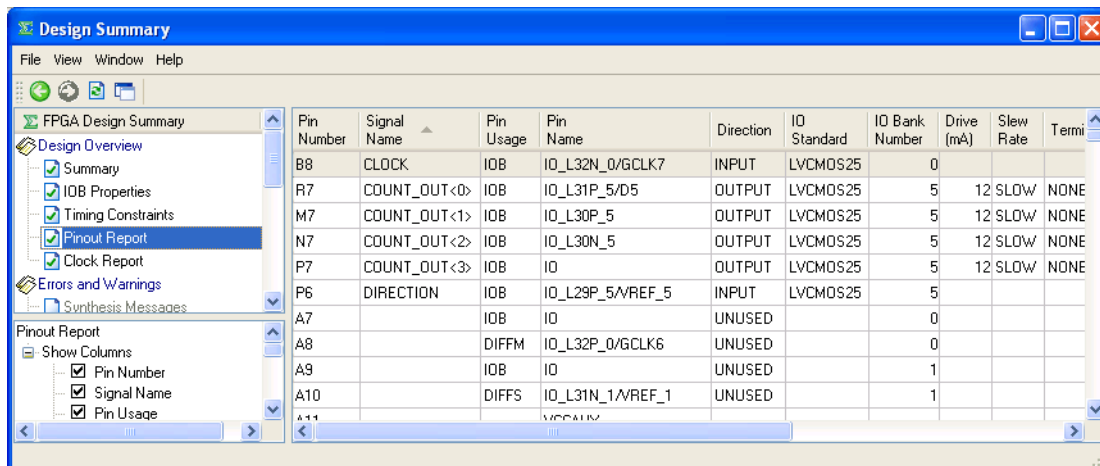
Reimplement Design and Verify Pin Locations

Reimplement the design and verify that the ports of the counter design are routed to the package pins specified in the previous section.

First, review the Pinout Report from the previous implementation by doing the following:

1. Open the Design Summary by double-clicking the **View Design Summary** process in the Processes window.

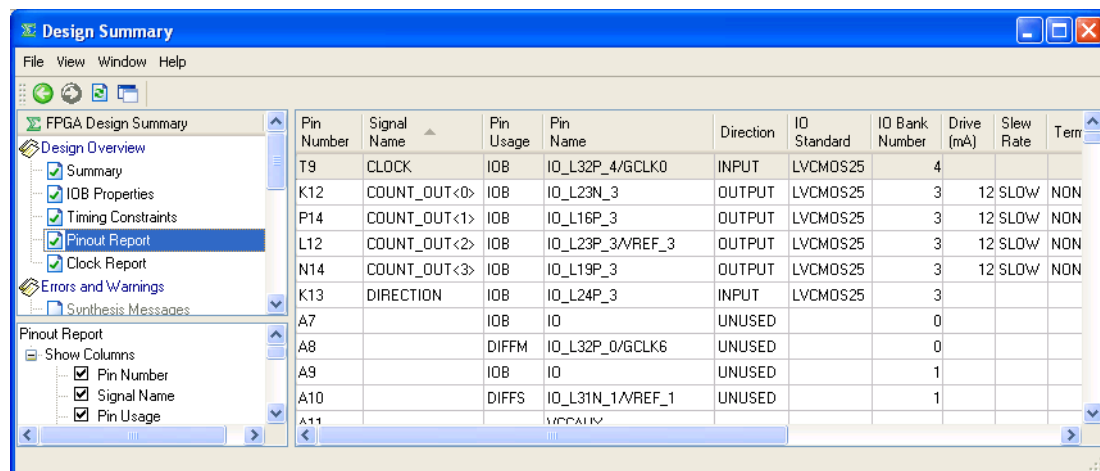
2. Select the **Pinout Report** and select the **Signal Name** column header to sort the signal names. Notice the Pin Numbers assigned to the design ports in the absence of location constraints.



Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termi
B8	CLOCK	IOB	IO_L32N_0/GCLK7	INPUT	LVCMS25	0			
R7	COUNT_OUT<0>	IOB	IO_L31P_5/D5	OUTPUT	LVCMS25	5	12 SLOW	NONE	
M7	COUNT_OUT<1>	IOB	IO_L30P_5	OUTPUT	LVCMS25	5	12 SLOW	NONE	
N7	COUNT_OUT<2>	IOB	IO_L30N_5	OUTPUT	LVCMS25	5	12 SLOW	NONE	
P7	COUNT_OUT<3>	IOB	IO	OUTPUT	LVCMS25	5	12 SLOW	NONE	
P6	DIRECTION	IOB	IO_L29P_5/REF_5	INPUT	LVCMS25	5			
A7		IOB	IO	UNUSED		0			
A8		DIFFM	IO_L32P_0/GCLK6	UNUSED		0			
A9		IOB	IO	UNUSED		1			
A10		DIFFS	IO_L31N_1/REF_1	UNUSED		1			
A11			VCCPIN						

Figure 21: Package Pin Locations Prior to Pin Location Constraints

3. Reimplement the design by double-clicking the **Implement Design** process.
4. Select the **Pinout Report** again and select the **Signal Name** column header to sort the signal names.
5. Verify that signals are now being routed to the correct package pins.



Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termi
T9	CLOCK	IOB	IO_L32P_4/GCLK0	INPUT	LVCMS25	4			
K12	COUNT_OUT<0>	IOB	IO_L23N_3	OUTPUT	LVCMS25	3	12 SLOW	NON	
P14	COUNT_OUT<1>	IOB	IO_L16P_3	OUTPUT	LVCMS25	3	12 SLOW	NON	
L12	COUNT_OUT<2>	IOB	IO_L23P_3/REF_3	OUTPUT	LVCMS25	3	12 SLOW	NON	
N14	COUNT_OUT<3>	IOB	IO_L19P_3	OUTPUT	LVCMS25	3	12 SLOW	NON	
K13	DIRECTION	IOB	IO_L24P_3	INPUT	LVCMS25	3			
A7		IOB	IO	UNUSED		0			
A8		DIFFM	IO_L32P_0/GCLK6	UNUSED		0			
A9		IOB	IO	UNUSED		1			
A10		DIFFS	IO_L31N_1/REF_1	UNUSED		1			
A11			VCCPIN						

Figure 22: Package Pin Locations After Pin Location Constraints

6. Close the Design Summary.

Verify Design using Timing Simulation

Use the same self-checking test bench waveform you created in the previous section to verify the counter design after it has been completely implemented. Timing simulation verifies that the design operates within the constraints specified after routing and logic delays are accounted for.

Run timing simulation as follows:

1. Select the **Post-Route Simulation** view from the drop-down list in the Sources window.
2. Select **counter_tbw** in the Sources window.
3. Run timing simulation by double-clicking the **Simulate Post-Place & Route Model** process found in the Xilinx ISE Simulator process group.

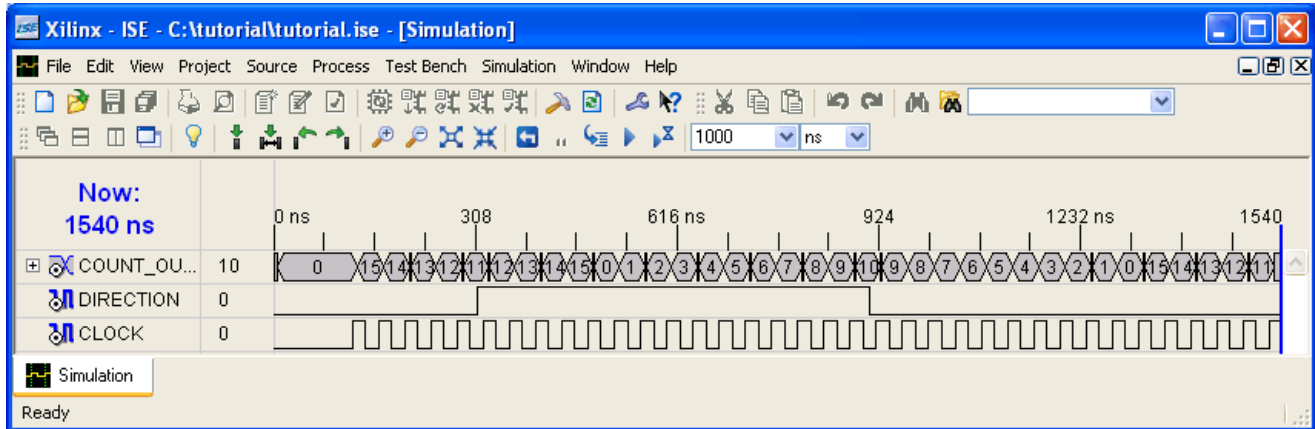


Figure 23: Post-Place & Route Simulation

4. Verify that the counter is counting up and down as directed by the stimulus on the DIRECTION port.
5. Verify that there are no ERRORS reported in the Simulator transcript window.
6. **Zoom in** to view the actual delay from the rising edge of CLOCK to a valid COUNT_OUT output change.
7. Close the simulation view.

You have completed timing simulation of your design using the ISE Simulator.

Download Design to the Spartan™-3 Demo Board

This is the last step in the design verification process. This section provides simple instructions for downloading the counter design to the Spartan-3 Starter Kit demo board.

1. Connect the 5V DC power cable to the power input on the demo board (J4).
2. Connect the download cable between the PC and demo board (J7).
3. Select **Synthesis/Implementation** from the drop-down list in the Sources window.
4. Select **counter** in the Sources window.
5. In the Processes window, click the “+” sign to expand the **Generate Programming File** processes.
6. Double-click the **Configure Device (iMPACT)** process.
7. The Xilinx WebTalk Dialog box may open during this process. Click **Decline**.
8. Select **Disable the collection of device usage statistics for this project only** and click **OK**.

iMPACT opens and the Configure Devices dialog box is displayed.

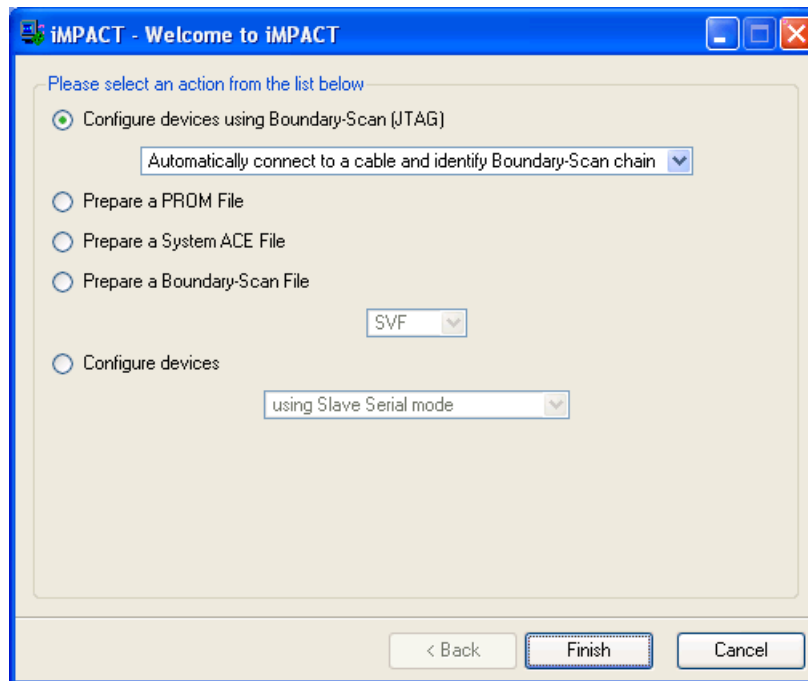


Figure 24: iMPACT Welcome Dialog Box

9. In the Welcome dialog box, select **Configure devices using Boundary-Scan (JTAG)**.
10. Verify that **Automatically connect to a cable and identify Boundary-Scan chain** is selected.
11. Click **Finish**.
12. If you get a message saying that there are two devices found, click **OK** to continue.
The devices connected to the JTAG chain on the board will be detected and displayed in the iMPACT window.

13. The **Assign New Configuration File** dialog box appears. To assign a configuration file to the xc3s200 device in the JTAG chain, select the counter .bit file and click **Open**.

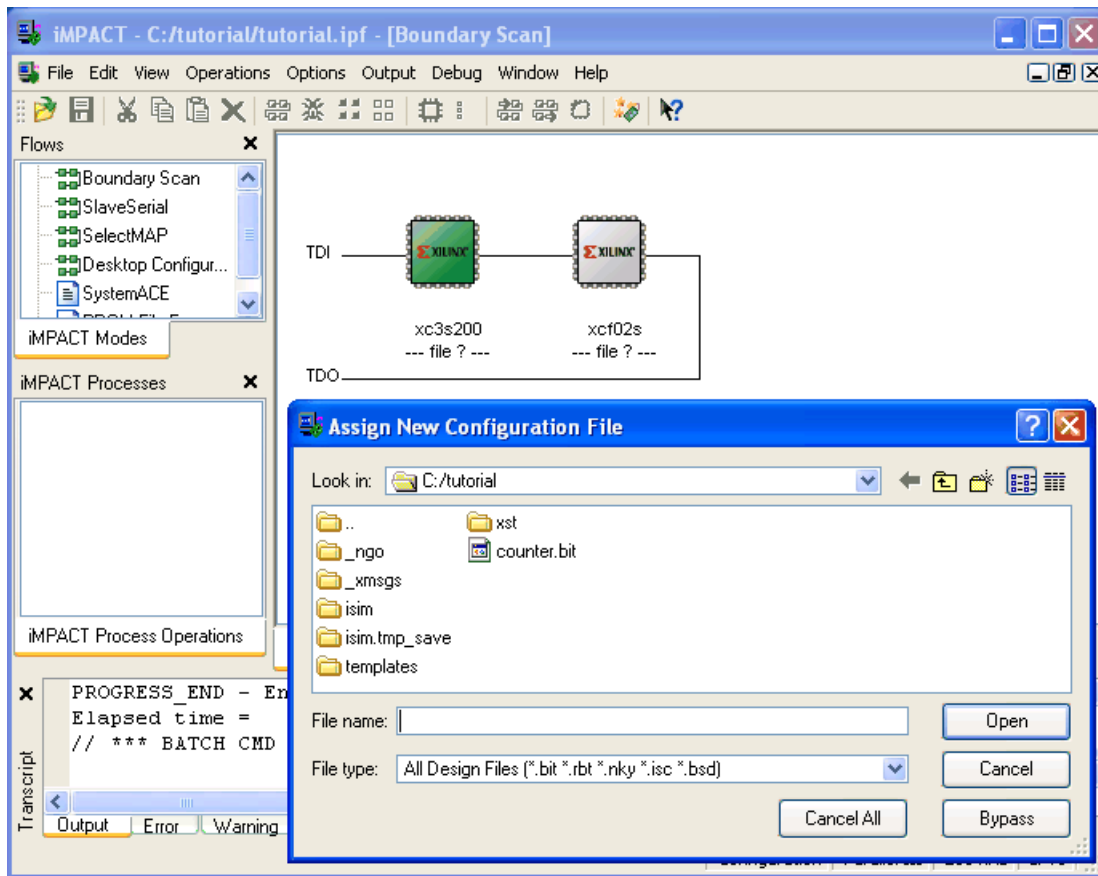


Figure 25: **Assign New Configuration File**

14. If you get a Warning message, click **OK**.
15. Select **Bypass** to skip any remaining devices.
16. Right-click on the xc3s200 device image, and select **Program...** The **Programming Properties** dialog box opens.
17. Click **OK** to program the device.

When programming is complete, the Program Succeeded message is displayed.

Program Succeeded

On the board, LEDs 0, 1, 2, and 3 are lit, indicating that the counter is running.

18. 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/>