

ISE 11.5 Quick Start Tutorial

(Created for CSE 141L)

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 Design Suite 11** → **ISE** → **Project Navigator**

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

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 Source Type: **HDL**
 - ◆ Synthesis Tool: **XST (VHDL/Verilog)**
 - ◆ Simulator: **ISE Simulator (VHDL/Verilog)**
 - ◆ Preferred Language: **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:

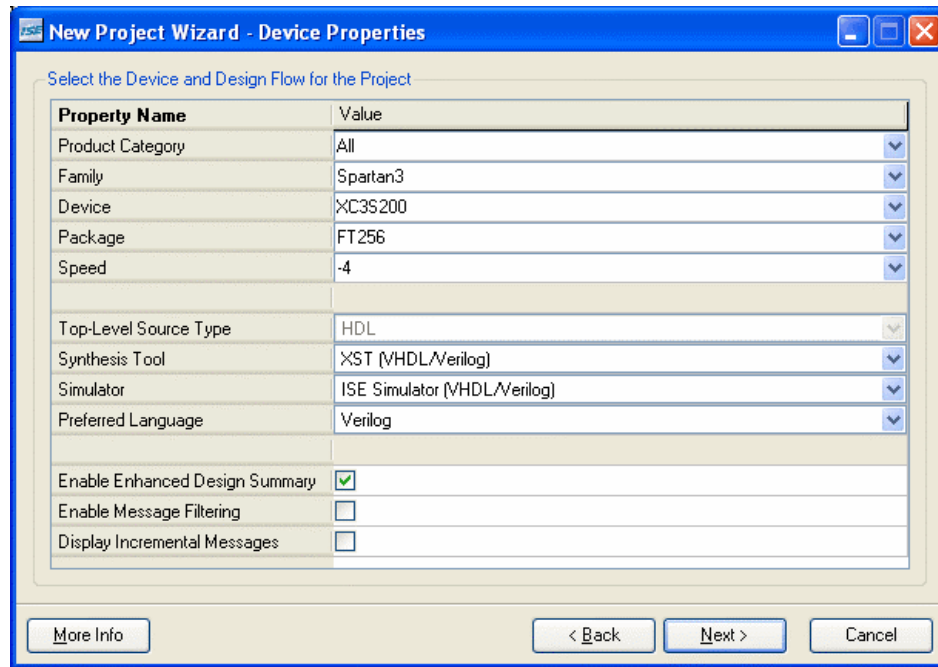


Figure 2: Project Device Properties

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 complete.

Create an HDL Source

In this section, you will create the top-level HDL file for your design.

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:

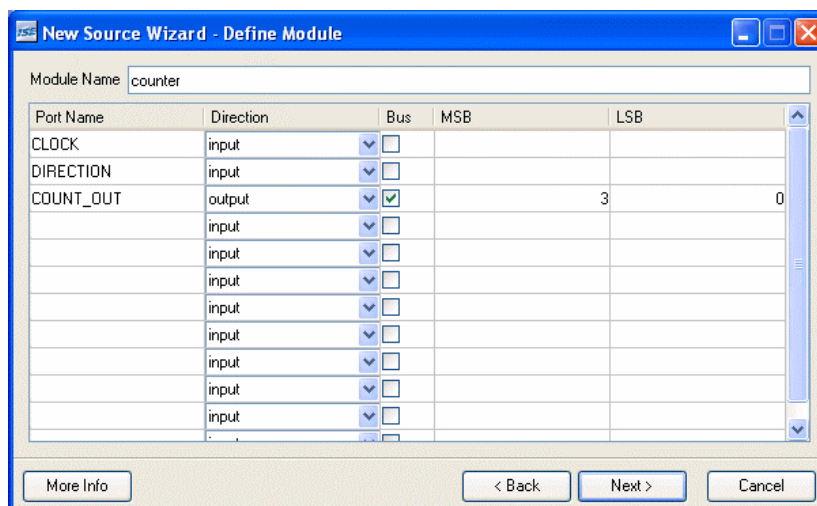


Figure 5: Define

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:

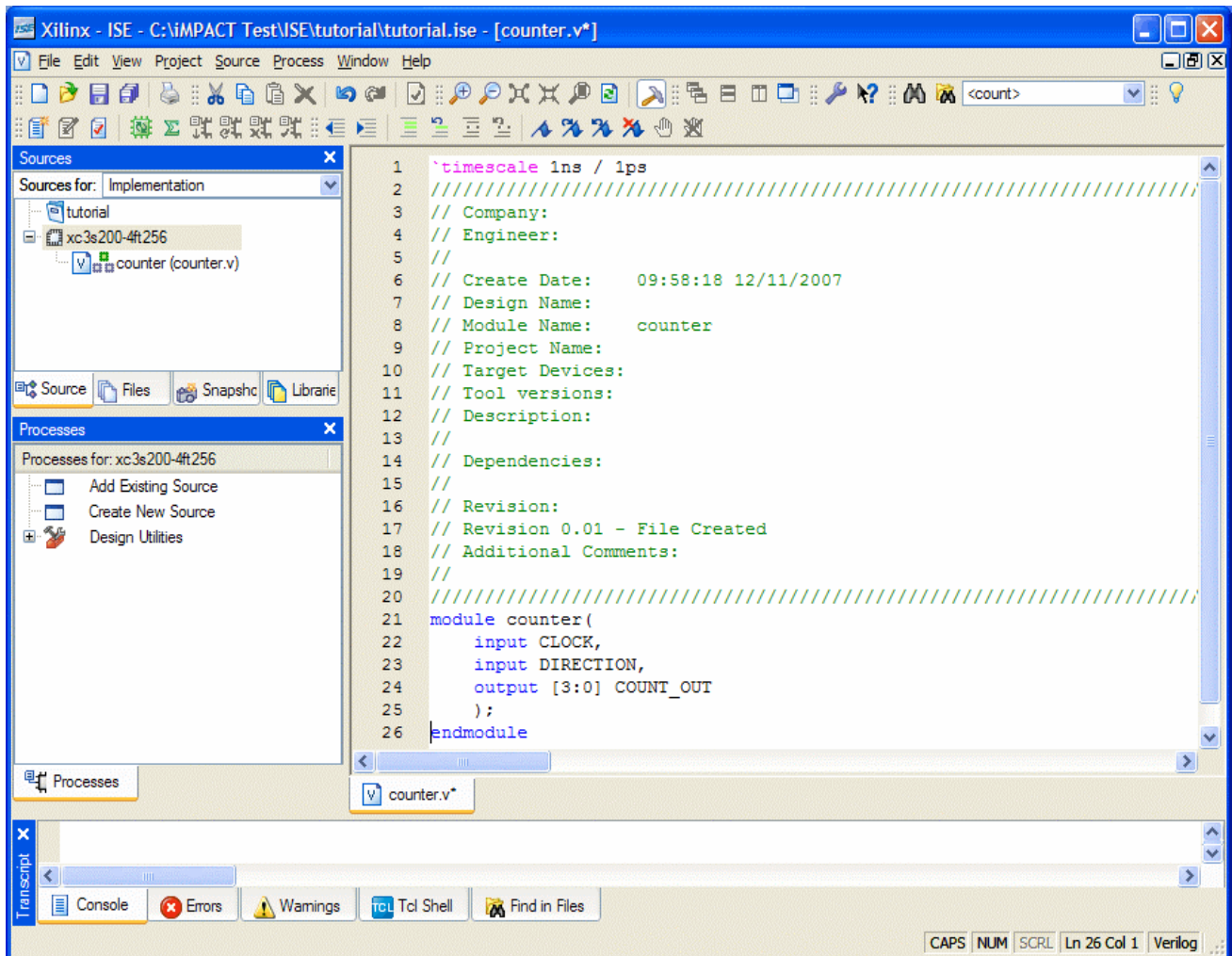


Figure 6: New Project in ISE

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** → **Counters** → **Binary** →

Up/Down Counters → Simple Counter

4. With Simple Counter selected, select **Right Click** → **Use in File, 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 **Implementation** is selected from the radio buttons 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.

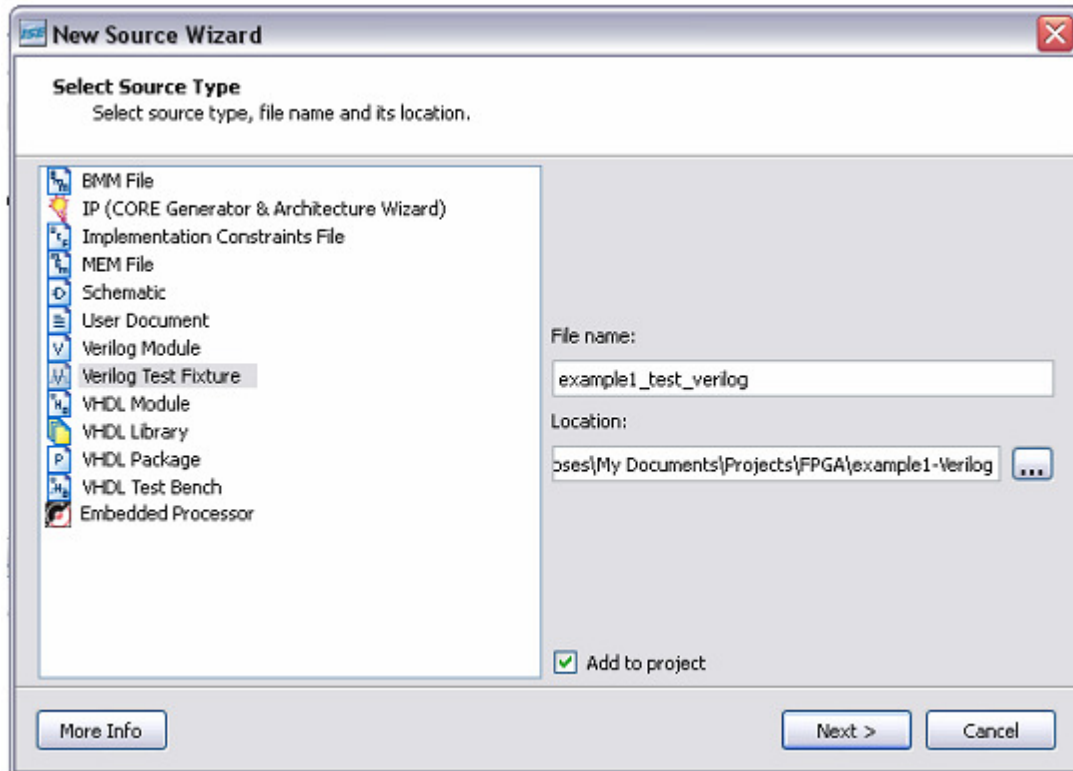
5. Close the HDL file.

Design Simulation

Verifying Functionality using Behavioral Simulation

Once the syntax is checked add a Verilog Test Fixture file to the project to run simulation.

1. Right click on the devices in the sources window and select **New Source...**
2. In the New Source Wizard select Verilog Test Fixture for the source type and enter and meaningful name for the file.



3. After clicking Next, the following dialog box asks you to select the source file you want to associate with the given test fixture file. This dictates which source file you actually run the simulation on. In this tutorial, we run the simulation on the top-level module of the design (counter.v). Click **Next** and **finish** to complete the creation
4. Select **Simulation** in the sources window to view the file.
5. Double click on the newly created testbench file. You will see that Xilinx has already generated lines of code to start the input definition. The code includes
 - a. A comment block template for documentation
 - b. A module statement
 - c. A UUT Instantiation
 - d. Input Instantiation

Scroll down the test fixture to see the code between 'initial begin' and 'end' blocks.

```

25 module tb;
26
27     // Inputs
28     reg CLOCK;
29     reg DIRECTION;
30
31     // Outputs
32     wire [3:0] COUNT_OUT;
33
34     // Instantiate the Unit Under Test (UUT)
35     counter uut (
36         .CLOCK(CLOCK),
37         .DIRECTION(DIRECTION),
38         .COUNT_OUT(COUNT_OUT)
39     );
40
41     initial begin
42         // Initialize Inputs
43         CLOCK = 0;
44         DIRECTION = 0;
45
46         // Wait 100 ns for global reset to finish
47         #100;
48
49         // Add stimulus here
50
51     end
52
53 endmodule
54

```

The simplest way of defining input stimulus in a Verilog test fixture is to use timing controls and delay, denoted by the pound symbol (#). For example, the statement #100 present in example1_test_verilog.v tells the simulator to delay for 100 ns. Therefore, any statement made after this timescale statement will occur after the 100 ns delay time. It's important to note that the timescale for the delay is defined by the `timescale statement at the beginning of the file. By default, the Xilinx tools define the timescale as 1ns/1ps, which indicates that the units are in nanoseconds while calculated time precision is 1 picoseconds.

To generate a clock signal that toggles every $CLOCKPERIOD = 40\text{ns}/2$ we define a always block

```

always
    #20 CLOCK = ~CLOCK;

```

6. Go to the processes window, expand the ISim Simulator (sic), and double-click Simulate Behavioral Model

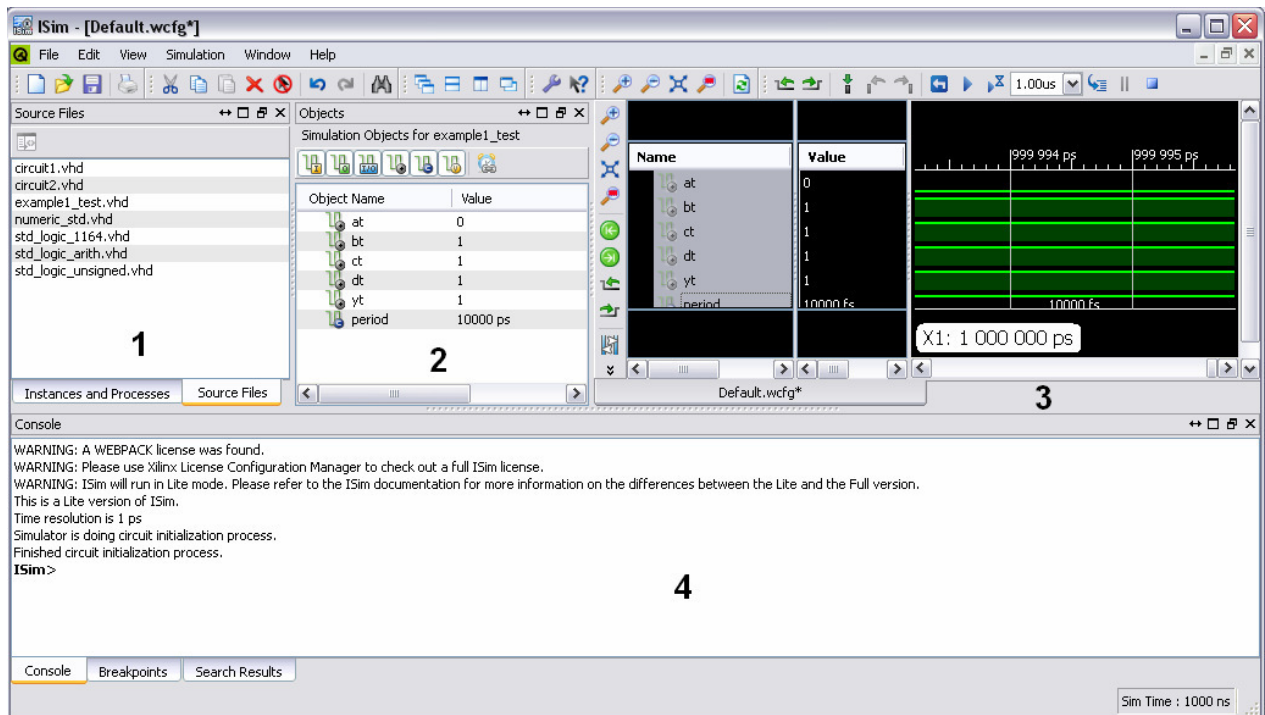
```

40
41     always
42         #20 CLOCK = ~CLOCK;
43
44     initial begin
45         // Initialize Inputs
46         CLOCK = 0;
47         DIRECTION = 0;
48
49         // Wait 100 ns for global reset to finish
50         #100;
51
52         // Add stimulus here
53     end
54
55     always
56     begin
57         #400 DIRECTION = 1;
58         #400 DIRECTION = 0;
59     end
60

```

ISE Simulator

Running the Simulate Behavioral Model process causes the ISim window to appear.



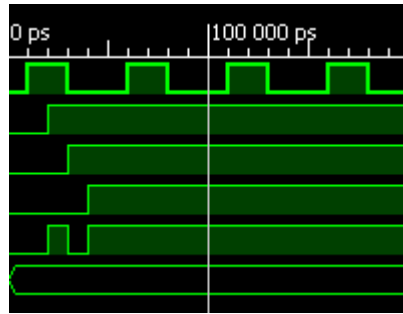
Some features of this window include:

1. a Source Files panel where source files to be viewed can be selected
2. an Objects panel where different signals can be added to the simulation
3. a simulation panel where the state of signals can be observed
4. a Console panel

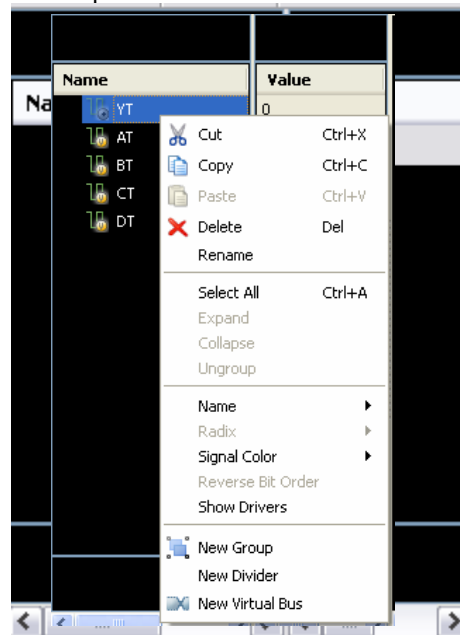
We first use the Zoom to Full View tool to see the full view of the simulation, which is located to the right of the magnifying glass on the simulation panel toolbar.



This displays the useful part of the simulation. Use the magnifying glass with the plus sign to zoom in further, as follows:



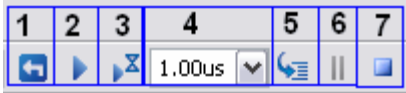
On the left side of the simulation panel there are columns labeled Name and Value:



For a given item on these columns, you can right-click and choose options to delete, rename, or change the color of the signal color.

You may also use the scroll bars to see the simulation at different times as well as observe more signals if you have a larger design.

The simulation control option on the top right side of the ISim toolbar contains the following features:



1. Restart simulation by stopping it and setting time back to 0.
2. Run simulation until all events are executed.
3. Run simulation for a specified time indicated by the Value box.
4. Amount of time and unit simulation is to run for.
5. Run simulation for one executable HDL instruction at a time.
6. Pause simulation.
7. Stop simulation.

Changing Stimulus

If you have different cases of stimulus that you wish to try out in the simulator, simply close ISim, edit the Verilog test fixture in ISE's text editor, and rerun the Simulate Behavioral Model process to open ISim again.

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 **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:

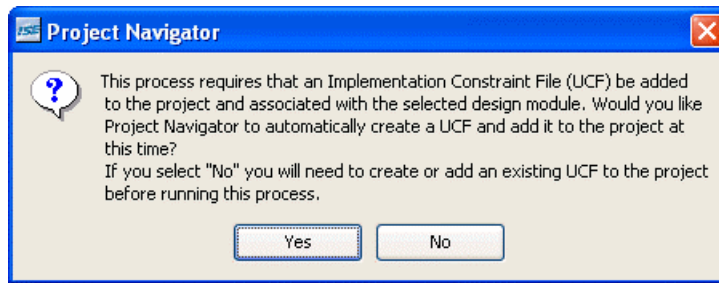


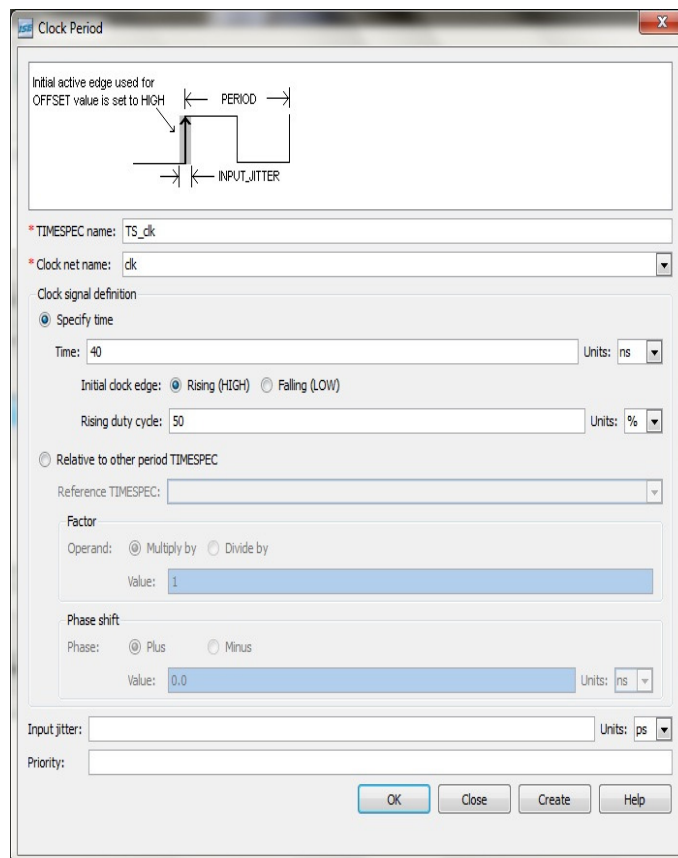
Figure 11: Prompt to Add UCF File to Project

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**.

5. In the Timing Constraints dialog, double click on **CLOCK** under unconstrained clocks windows. Enter the period as 40.



6. Click **Create** and **Ok**.

After the information has been entered, the dialog should look like what is shown below.

TIMESPEC Name * ^	Clock Time Name	Clock Net *	Reference TIMESPEC	Period	Duty Cycle	Factor	Edge	Phase Shift	Input Jitter	Source
1 TS_CLOCK	CLOCK	CLOCK		40 ns	50 %		HIGH			co...cf
2										

Figure 12: Creating Timing Constraints

7. Select **Timing Constraints** under Constraint Type in the Timing Constraints tab and the newly created timing constraints are displayed.
8. Save the timing constraints. If you are prompted to rerun the TRANSLATE or XST step, click **OK** to continue.
9. 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.

The screenshot shows the Xilinx ISE Design Summary window for a project named 'tutorial'. The window is divided into several panes:

- Sources:** Shows the source files for the implementation, including 'counter - Behavioral (counter.vhd)'.
- Processes:** Shows the implementation process 'Implement Design' with a green checkmark, indicating successful completion.
- Design Summary:** Provides a detailed overview of the design, including project status, partition summary, and device utilization summary.

tutorial Project Status (12/11/2007 - 11:35:57)

Field	Value	Current State	Details
Project File:	tutorial.ise	Placed and Routed	
Module Name:	counter	• Errors:	No Errors
Target Device:	xc3s200-4ft256	• Warnings:	No Warnings
Product Version:	ISE 10.1 - Foundation Simulator	• Routing Results:	All Signals Completely Routed
Design Goal:	Balanced	• Timing Constraints:	All Constraints Met
Design Strategy:	Xilinx Default (unlocked)	• Final Timing Score:	0 (Timing Report)

tutorial Partition Summary

No partition information was found.

Device Utilization Summary

Logic Utilization	Used	Available	Utilization	Note(s)
Number of Slice Flip Flops	4	3,840	1%	
Number of 4 input LUTs	4	3,840	1%	
Logic Distribution				

The Transcript window at the bottom shows the message: "Process 'Generate Post-Place & Route Static Timing' completed successfully".

Figure 14: Post Implementation Design Summary

5. Locate the **Performance Summary** table near the bottom of the Design Summary.
6. 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	Check	Worst Case Slack	Best Case Achievable	Timing Errors	Timing Score
1 Yes	TS_CLOCK = PERIOD TIMEGRP "CLOCK" 40 ns HIGH 50%	SETUP HOLD	36.829ns	3.171ns	0	0
			0.998ns		0	0

Figure 15: All Constraints Met Report

7. 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:

1. Verify that **counter** is selected in the Sources window.
2. Double-click the **I/O Pin Planning(Plan Ahead) – Post Synthesis** process found in the User Constraints process group. The **Xilinx Plan Ahead Tool** opens.
3. In the IO ports tab expand the **Scalar ports**.
4. Select **CLOCK** and type **T9** in the site field in the I/O ports properties tab. Click on **Apply** to confirm the changes.
5. Similarly apply **K13** as location for **DIRECTION**
6. Expand the **COUNT_OUT (4)** in the IO ports tab and then select of the ports to apply constraints. **COUNT_OUT[3]** to Pin **N14**, **COUNT_OUT[2]** to pin **L12**, **COUNT_OUT** to pin **P14** and **COUNT_OUT[0]** to pin **K12**.

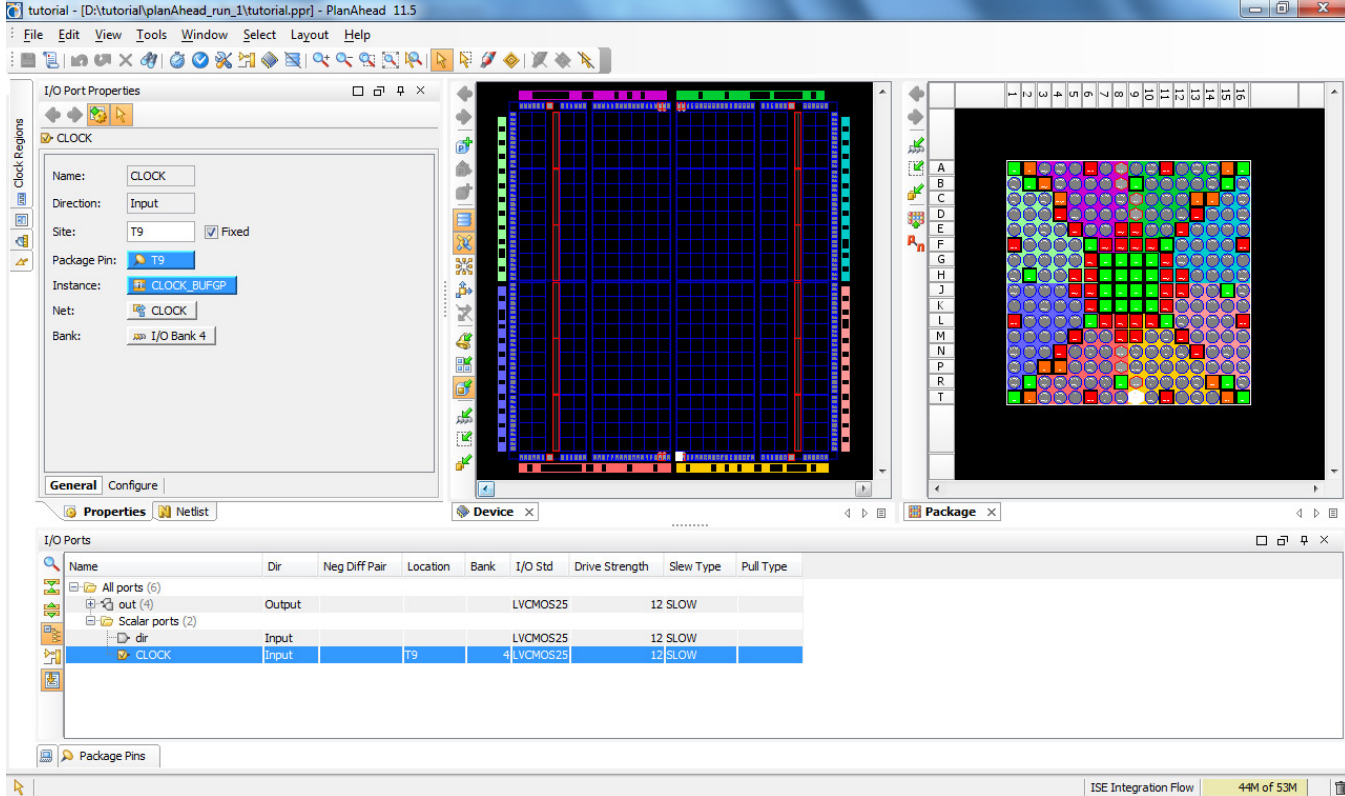


Figure 16: Package Pin Locations

5. Select **File** → **Save**.
6. Close Plan Ahead.

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.

The screenshot shows the 'Design Summary' window with the 'Pinout Report' selected. The report is sorted by Signal Name. The following table represents the data shown in the report:

Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank	Drive Number	Drive (mA)	Slew Rate	Termi
B8	CLOCK	IOB	IO_L32N_0/GCLK7	INPUT	LVC MOS25	0				
R7	COUNT_OUT<0>	IOB	IO_L31P_5/D5	OUTPUT	LVC MOS25	5	12	SLOW	NONE	
M7	COUNT_OUT<1>	IOB	IO_L30P_5	OUTPUT	LVC MOS25	5	12	SLOW	NONE	
N7	COUNT_OUT<2>	IOB	IO_L30N_5	OUTPUT	LVC MOS25	5	12	SLOW	NONE	
P7	COUNT_OUT<3>	IOB	IO	OUTPUT	LVC MOS25	5	12	SLOW	NONE	
P6	DIRECTION	IOB	IO_L29P_5/REF_5	INPUT	LVC MOS25	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		VCCALN								

Figure 17: Package Pin Locations Prior to Pin Location Constraints

- Reimplement the design by double-clicking the **Implement Design** process.
- Select the **Pinout Report** again and select the **Signal Name** column header to sort the signal names.
- 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	Term
T9	CLOCK	IOB	ID_L32P_4/GCLK0	INPUT	LVC MOS25		4		
K12	COUNT_OUT<0>	IOB	ID_L23N_3	OUTPUT	LVC MOS25	3	12	SLOW	NON
P14	COUNT_OUT<1>	IOB	ID_L16P_3	OUTPUT	LVC MOS25	3	12	SLOW	NON
L12	COUNT_OUT<2>	IOB	ID_L23P_3/REF_3	OUTPUT	LVC MOS25	3	12	SLOW	NON
N14	COUNT_OUT<3>	IOB	ID_L19P_3	OUTPUT	LVC MOS25	3	12	SLOW	NON
K13	DIRECTION	IOB	ID_L24P_3	INPUT	LVC MOS25		3		
A7		IOB	ID	UNUSED			0		
A8		DIFFM	ID_L32P_0/GCLK6	UNUSED			0		
A9		IOB	ID	UNUSED			1		
A10		DIFFS	ID_L31N_1/REF_1	UNUSED			1		
A11			ID	UNUSED					

Figure 18: Package Pin Locations After Pin Location Constraints

- Close the Design Summary.