Introduction to Xilinx Tools

Using the Lab Machines **2**

Using Xilinx ISE **2**

Step 1: Project Setup 2

Step 2: Module Setup 6

Step 3: Synthesis 9

Step 4: Testbench 10

Appendix **14**

A1: Templates 14

Using the Lab Machines

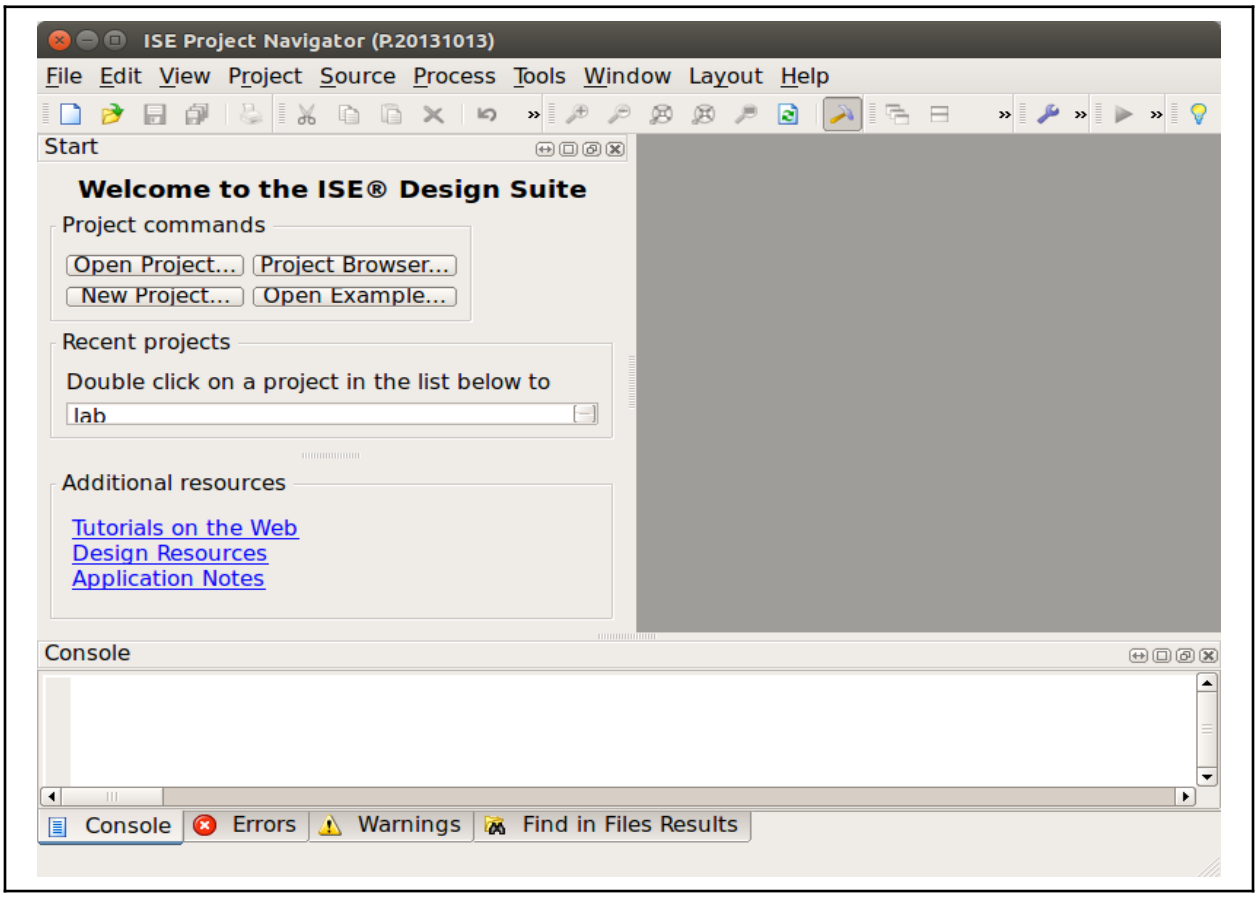
Welcome to the world of designing digital systems. Today you will get an introduction to the Xilinx ISE software. You will be implementing a simple digital system (`full_adder`) to get you familiar with the tools. To open Xilinx ISE on a lab machine, first login and open a terminal and type `ise`.

Using Xilinx ISE

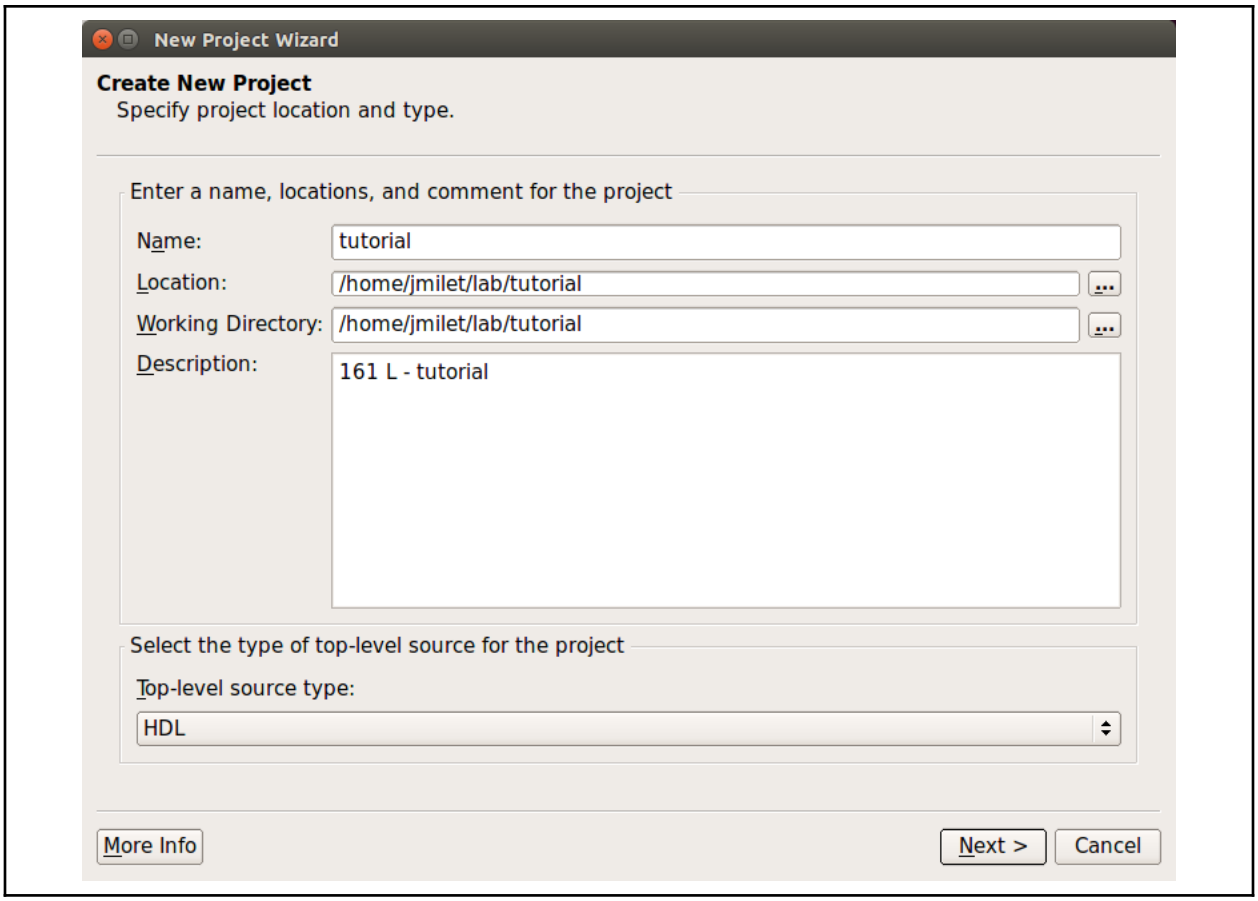
This tutorial will help you become familiar with using Xilinx ISE to develop on a Xilinx based FPGA board.

Step 1: Project Setup

1. Open Xilinx ISE Design Suite (login in, open terminal, type: `ise`)
2. Select a "`New Project`". Note: If you are not prompted to create a new project, start the new project wizard by selecting **New Project** in the **File** menu.

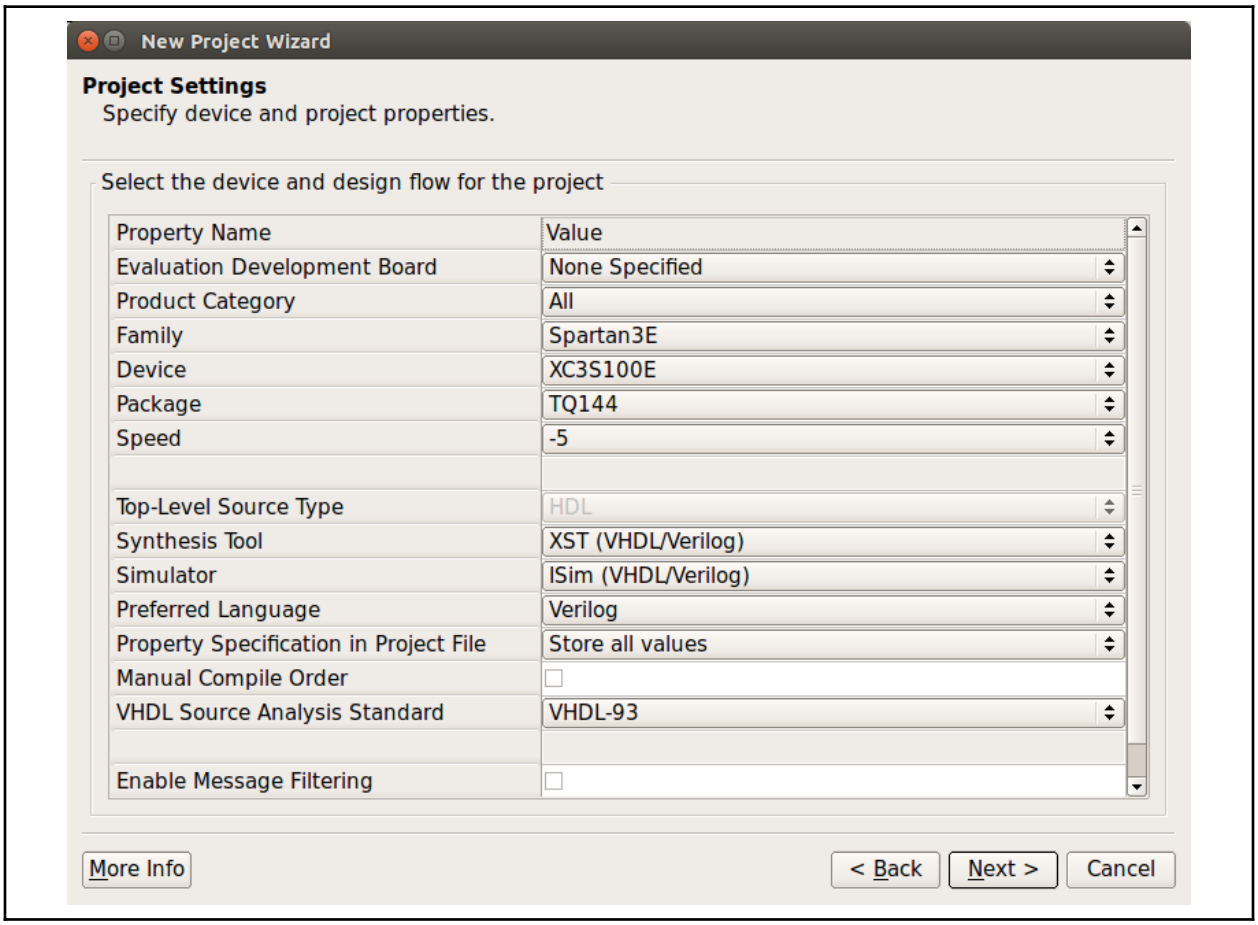


3. Enter the desired project name as well as the working directory. Also enter a description of the Project. Then Select **Next >**

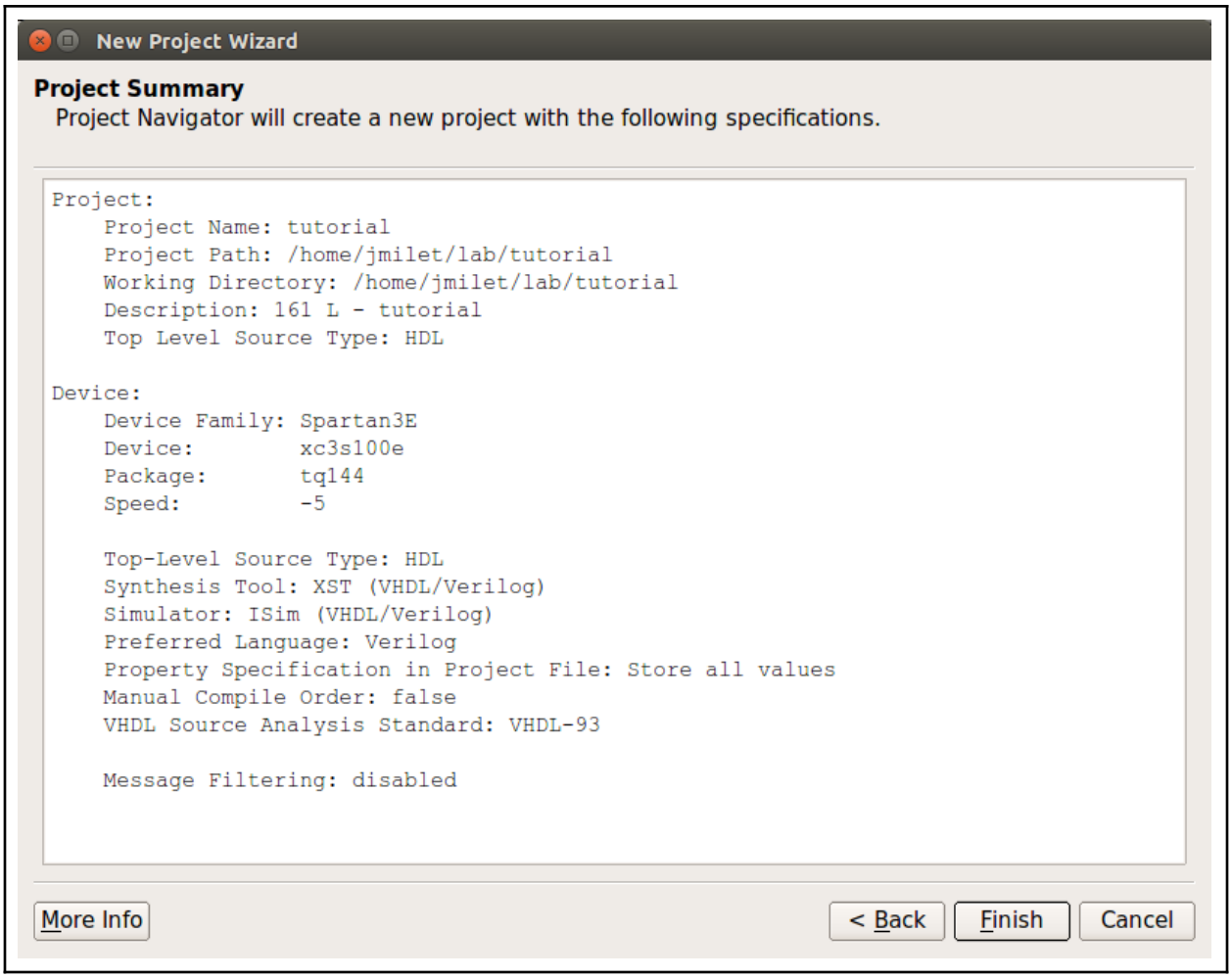


4. Select the Evaluation Development Board and Device, if known. (I.E. Spartan-3E Starter Board). Otherwise, you will have to look at the FPGA physical board chip or product documentation for the Family, Device, Package, and Speed. Other settings are shown below.

Synthesis Tool: XST (VHDL/Verilog)
Simulator: ISim (VHDL/Verilog)
Preferred Language: Verilog
Manual Compile Order: Unselected
VHDL Source Analysis Standard: VHDL-93
Enable Message Filtering: Unselected

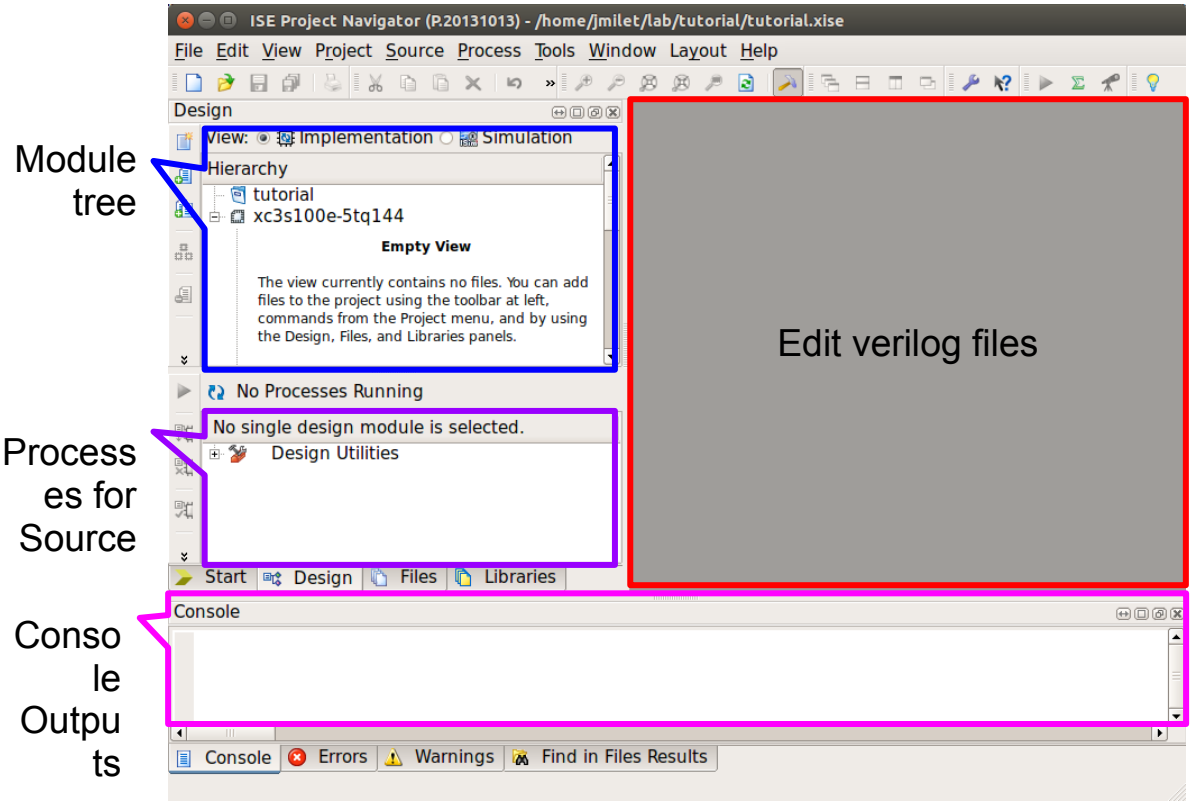


5. Select **Next** >. A Project Summary will then be shown. Review the content for correctness and select **Finish** if correct, otherwise repeat the previous steps.



Step 2: Module Setup

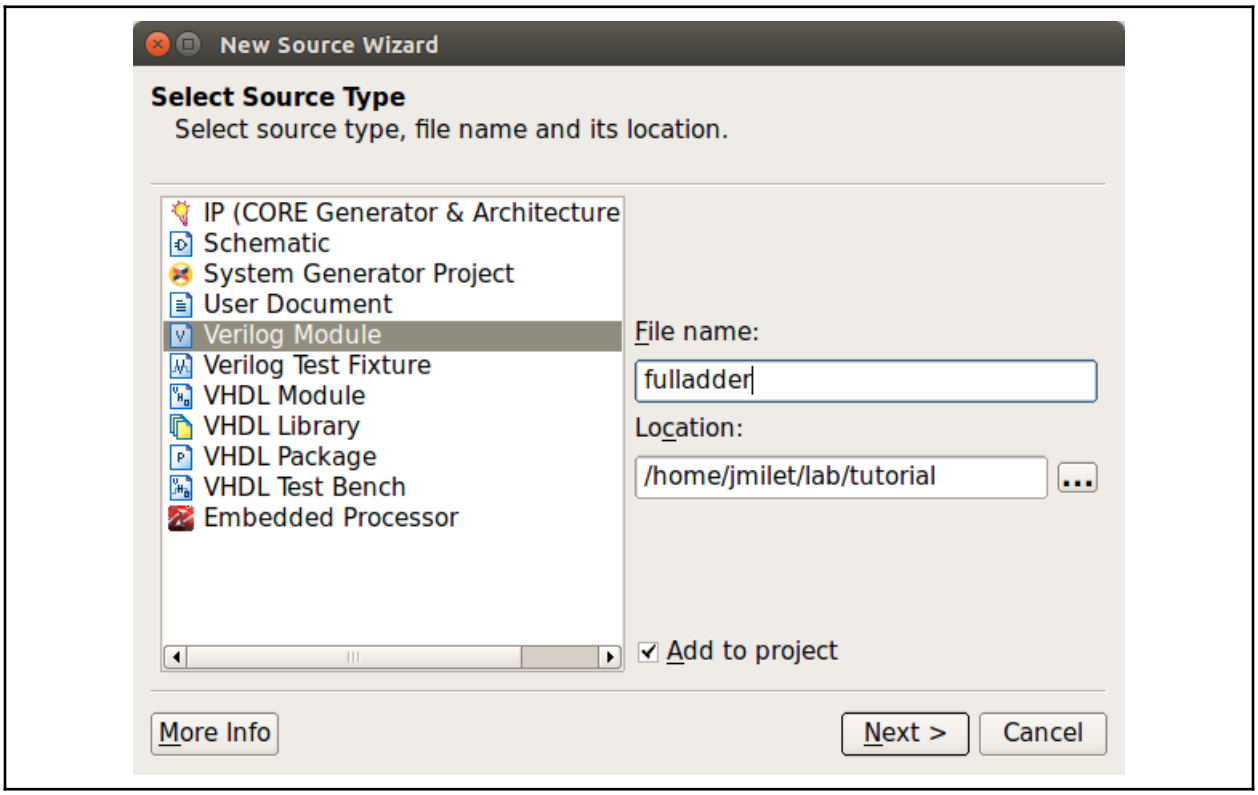
You are now viewing the Project Navigator. The Sources in the Project section will automatically organize your **Verilog module tree** (Top Left). The **Processes for Source** panel will allow you to perform various processes such as synthesis or device programming, view reports, and access useful tools (Middle Left). The bottom panel contains **console outputs** including errors, warnings and find results; these are useful while debugging (bottom section). The panel on the right is used to **edit** any files or documents you have opened.



Now you will create a new Verilog module.

1. Right-Click on the chip icon and select **New Source**. *Note:* that you can add already created sources.
2. Select the option **Verilog Module** in the left panel.

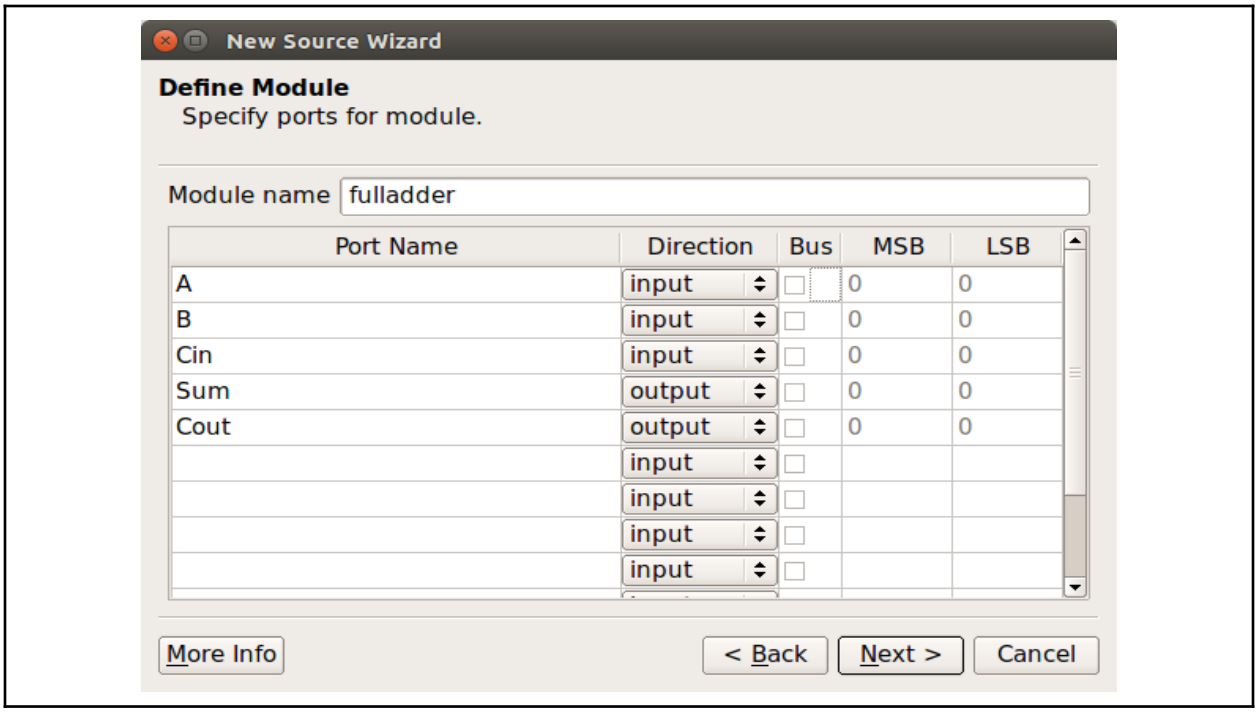
3. Type the name of the new module, `fulladder` in this case. Then select **Next >**



4. In the New Source Wizard dialog, the interface (inputs and outputs) of the module can be specified. In the example we have specified the module has 3 inputs (`A`, `B`, `Cin`) and 2 outputs (`Sum` and `Cout`). All the signals are one bit, so there is no need to check the `Bus` checkbox. If bus signals were to be required (multiple bits), we would check the

`Bus` checkbox and enter the number of bits in the bus. Here MSB and LSB refers to the most and least significant bit. Then select **Next >** .

5. A summary of the newly created module is shown. To continue select **Finish >** .



Step 3: Synthesis

Now, it is time to specify the internals of the `fulladder` module shown below. Type the following into the edit window for the file.

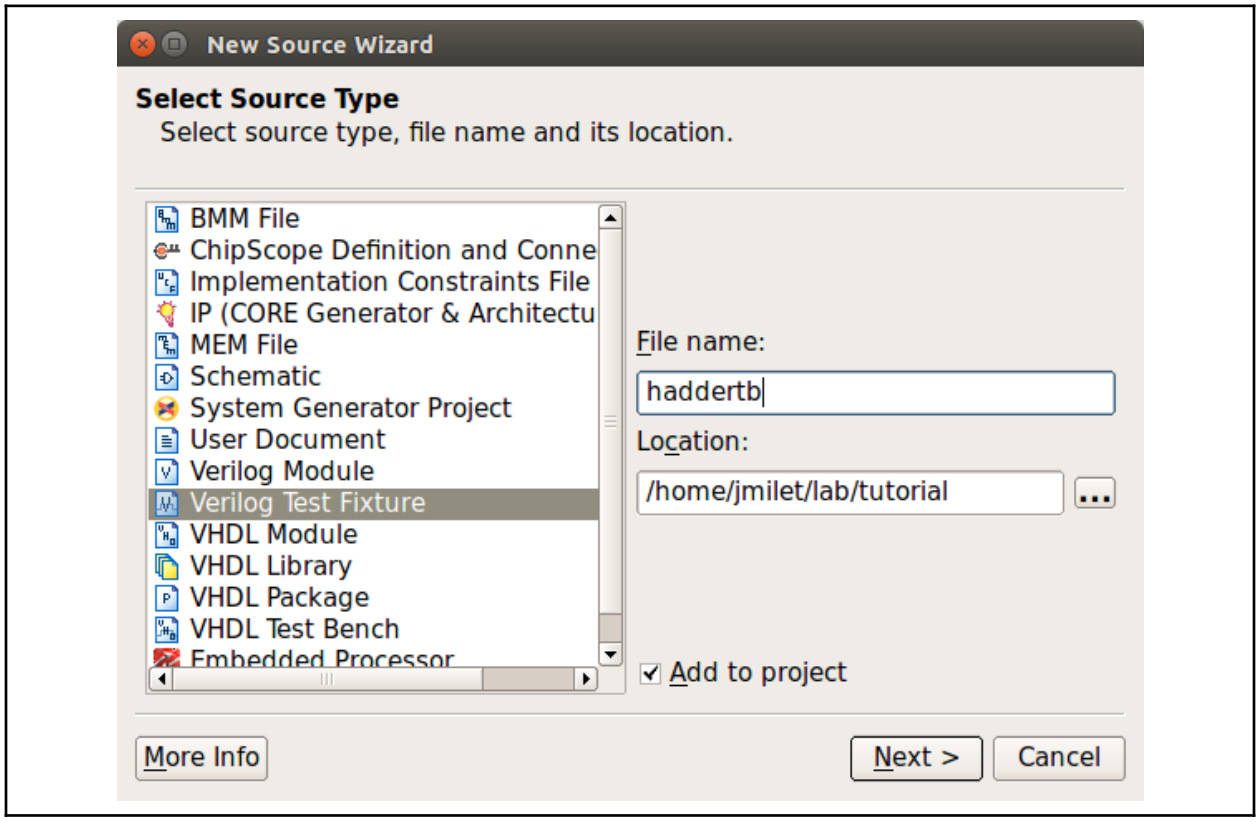
```
module fulladder (
      input A,
      input B,
      input Cin,
      output Sum,
      output cout
);

wire s1, t1, t2, t3;

xor x1 (s1, A, B);
xor x2 (Sum, s1, Cin);
and a1 (t3, A, B);
and a2 (t2, A, Cin);
and a3 (t1, B, Cin);
or o1 (Cout, t1, t2, t3);

endmodule
```

Next, select the option **Synthesize - XST** in the processes panel. This option checks that the code is correct and synthesizable. If warnings or errors occur, you should review and correct the verilog code before proceeding.

Step 4: Testbench

Now, we are going to add a new module in order to test the `fulladder` above.

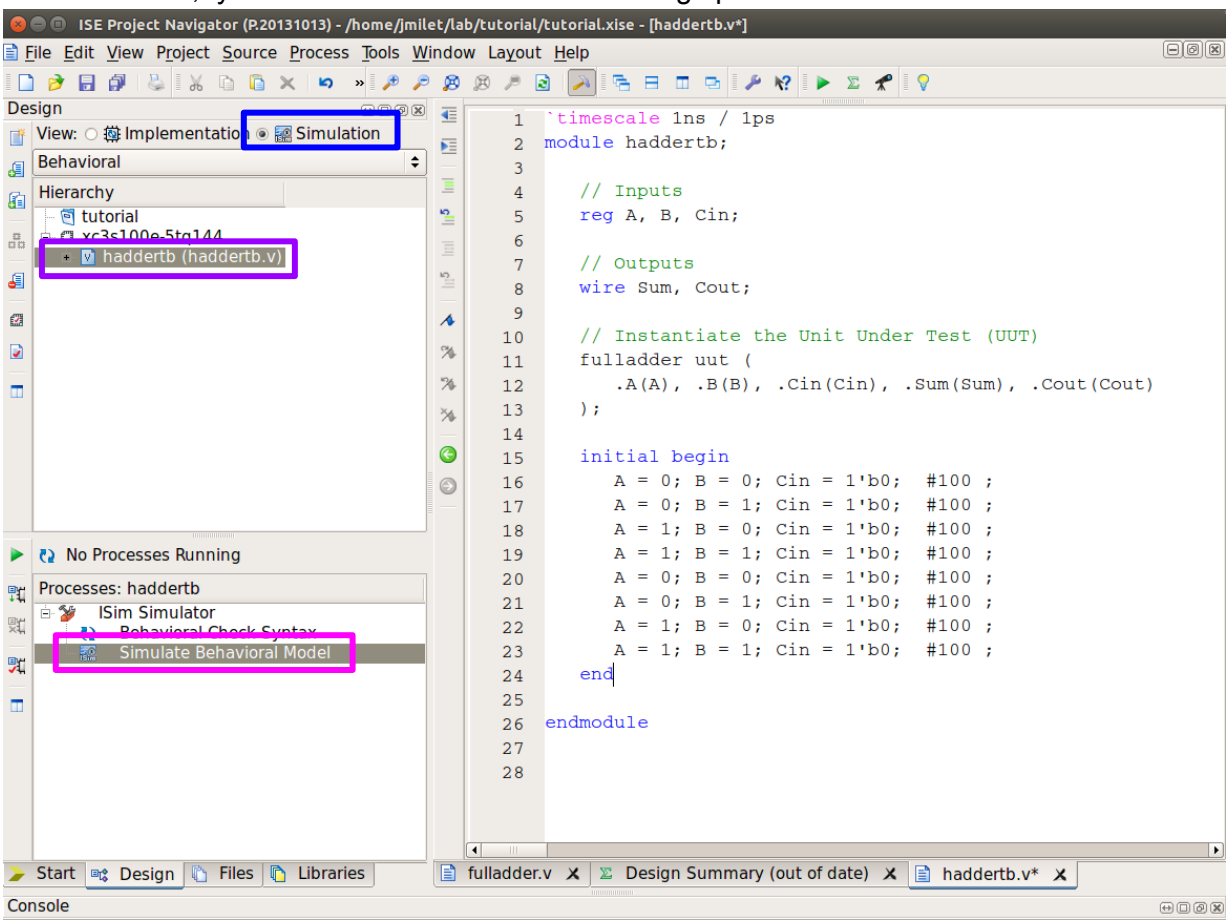
1. Add a new source
2. Select Verilog Test Fixture
3. In the File Name box, type the name testbench name (`fulladder_tb`). Select **Next > .**
4. In the dialog **Associate Source** select the module under test (`fulleradder`). Select **Next > .**
5. A summary shows up, now select **Finish >** .



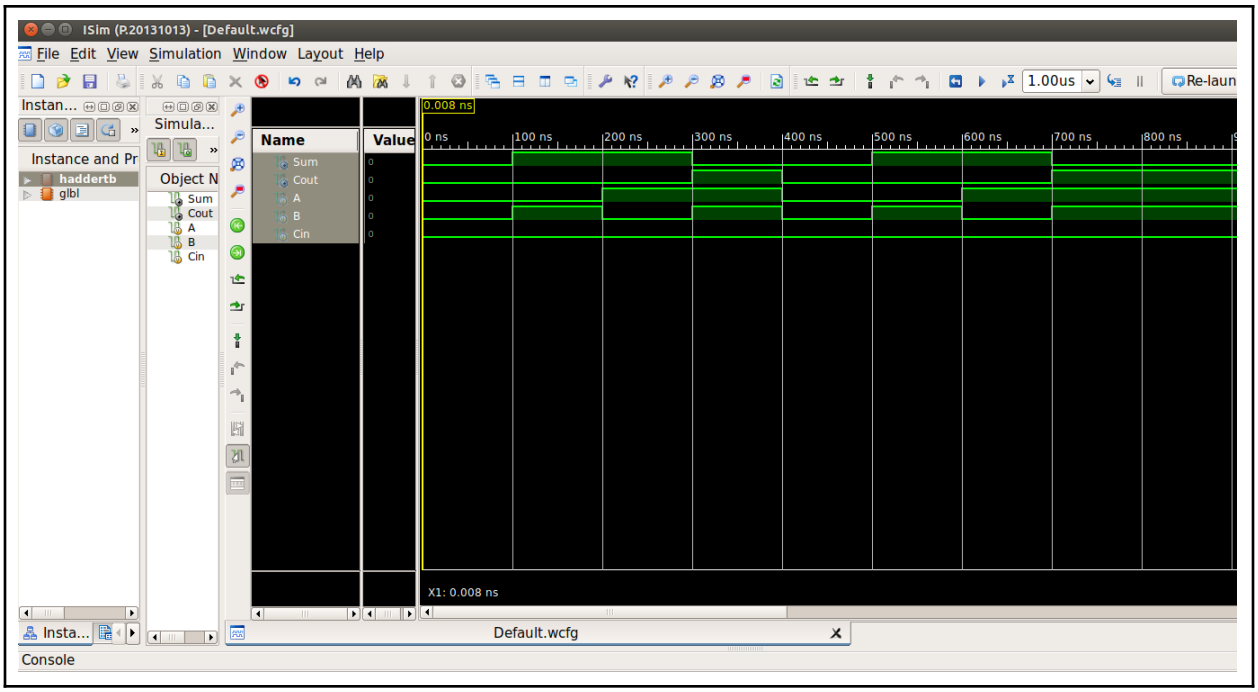
6. Now, enter the verilog specification of the testbench has shown in the following dialog.

```
`timescale 1ns / 1ps
module fulladder_tb;

// inputs
reg A, B, Cin;

// outputs
wire Sum, Cout;

// Instantiate the Unit Under Test (uut)
fulladder uut (
      .A(A), .B(B), .Cin(Cin), .Sum(Sum), .Cout(Cout)
);

intial begin
      A = 0; B = 0; Cin = 1'b0; #100;
      A = 0; B = 1; Cin = 1'b0; #100;
      A = 1; B = 0; Cin = 1'b0; #100;
      A = 1; B = 1; Cin = 1'b0; #100;
      A = 0; B = 0; Cin = 1'b1; #100;
      A = 0; B = 1; Cin = 1'b1; #100;
      A = 1; B = 0; Cin = 1'b1; #100;
      A = 1; B = 1; Cin = 1'b1; #100;
end
endmodule
```

7. Next, in the design panel check the **simulation box**. Select the **`fulladder_tb`** and execute the **Simulate Behavioral Model** from the Process panel on the bottom left. If errors, you have to double check the verilog specification of the testbench.



8. In the Processes from the Source panel double click the Simulate Behavioral Model. A new window will appear showing the simulation. If the test bench completes with no errors in the command prompt then simulation was successful for the tested cases. You might have to zoom out to view the signals in *ns*. Zoom out by pressing F7 and zoom in by pressing F8 or use the shortcuts on the top bar. **Note:** the testbench does not necessarily run to the final wait statement. You might need to run the test until you get the message "Note: Done with testbench".

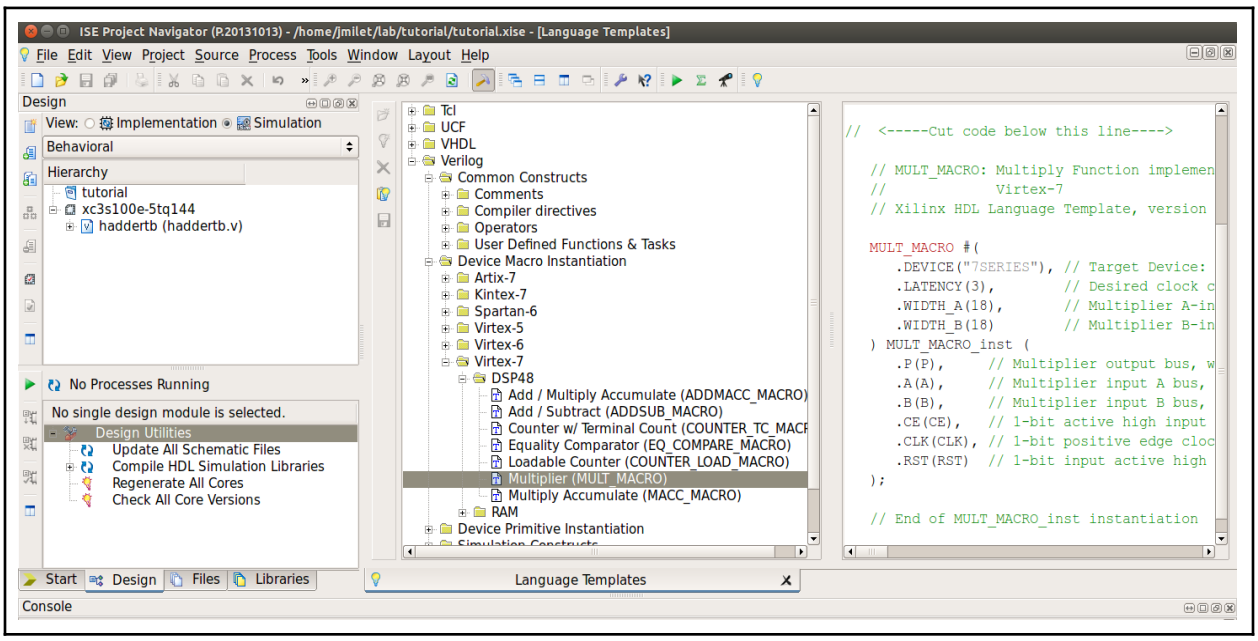


You can also select the signals and change the display format from binary to hexadecimal or decimal by selecting radix in the right-click menu of a signal. Other functionality includes: being able to add signals to the test bench from internal signals by dragging signals/variables from the center pane to the rightmost pane. Other components can be explored by expanding and selecting object in the left most pane. Switch back to the implementation view to continue editing.

Appendix

A1: Templates

Templates are a very useful resource for engineers. Unlike many other programming languages, only a subset of Verilog can be synthesized to hardware implementation. To access the language templates: go to Edit > Language Templates.



In viewing the template look only at Verilog > Synthesis Constructs > Coding Examples when design hardware. These templates will give you an intuition for how to create hardware structures.