Oregon TECH

Computer Systems Engineering Technology
CST 345—HW/SW Co-Design

Lab 01 – Xilinx Design Flow and Tools
Winter 2015
Instructor: Troy Scevers
Possible Points: 10

Name_____
Due Date: Monday, January 12th @ 5pm

Instructions

This lab is an introduction to hardware design using Verilog-HDL along with the Xilinx ISE 14.7 ISE tools. No new logic design concepts are presented in this lab. The goal of this lab is for you to become familiar with the tools you will be using for the rest of the quarter:

- The Xilinx **ISE 14.7** Project Navigator.
- The Digilent Nexys 3 board

Consider this lab a “no-brainer” warm up for the next labs. In previous courses, I have heard students lament, “I wish I had paid more attention to lab one…” Please read carefully, pay attention, and take your time.

Procedure

Project Navigator Overview

Project Navigator is divided into four sub-areas, as seen in Figure 1. On the top left is the Sources window which hierarchically displays the elements included in the project for the currently selected stage of the design flow. Beneath the Sources window is the Processes window which displays available processes for the currently selected source. The third area at the bottom is the Console window which displays status messages, errors, and warnings. The fourth area, on the top right, is for viewing and editing project files. Each window may be resized or moved within Project Navigator.

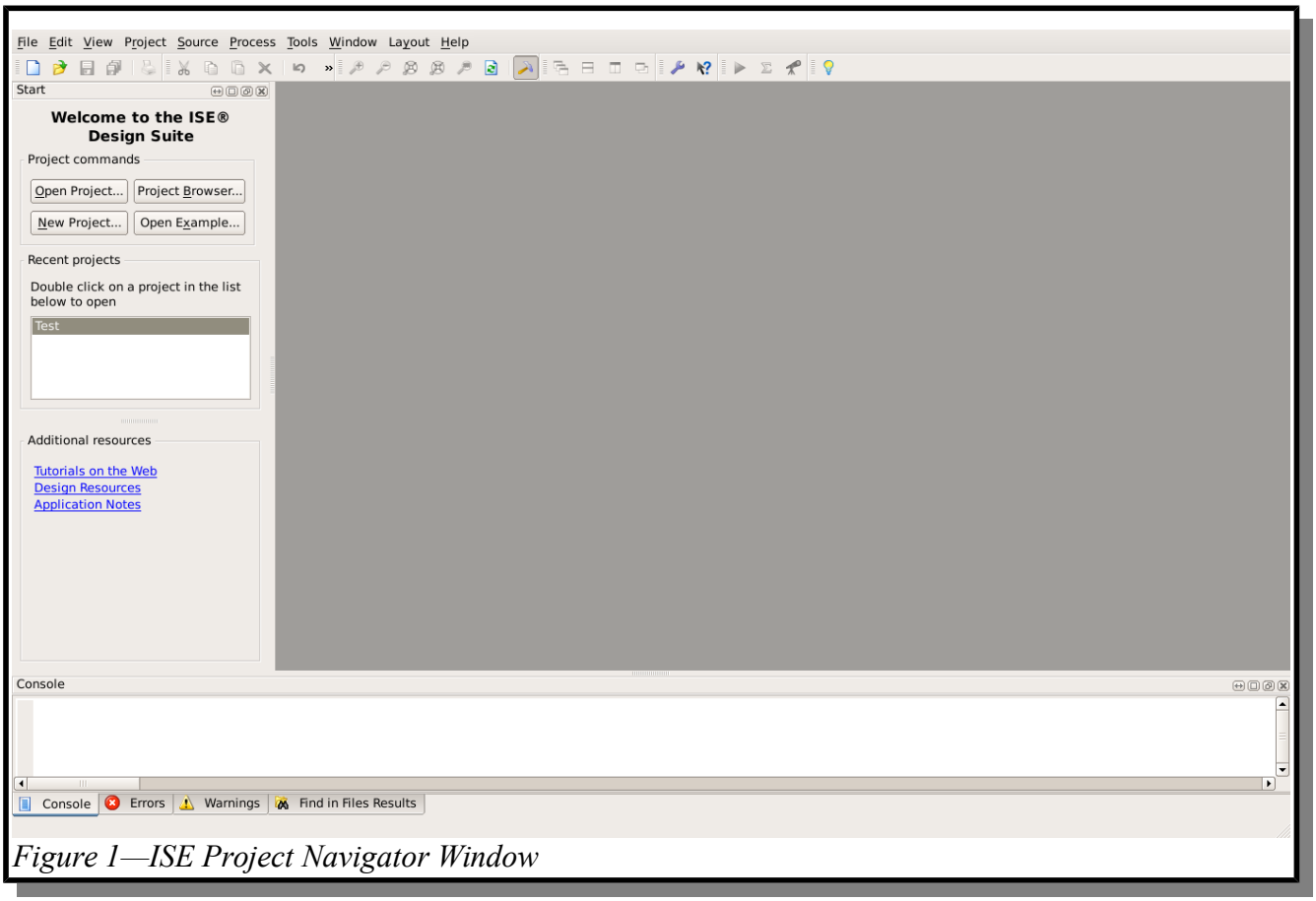


Figure 1—ISE Project Navigator Window

Design Entry

The design used in this tutorial is a simple adder. The design will be described in Verilog-HDL. Select the <New Project> button from the Project Command window. The first of the New Project dialog boxes will appear, as shown in Figure 2.

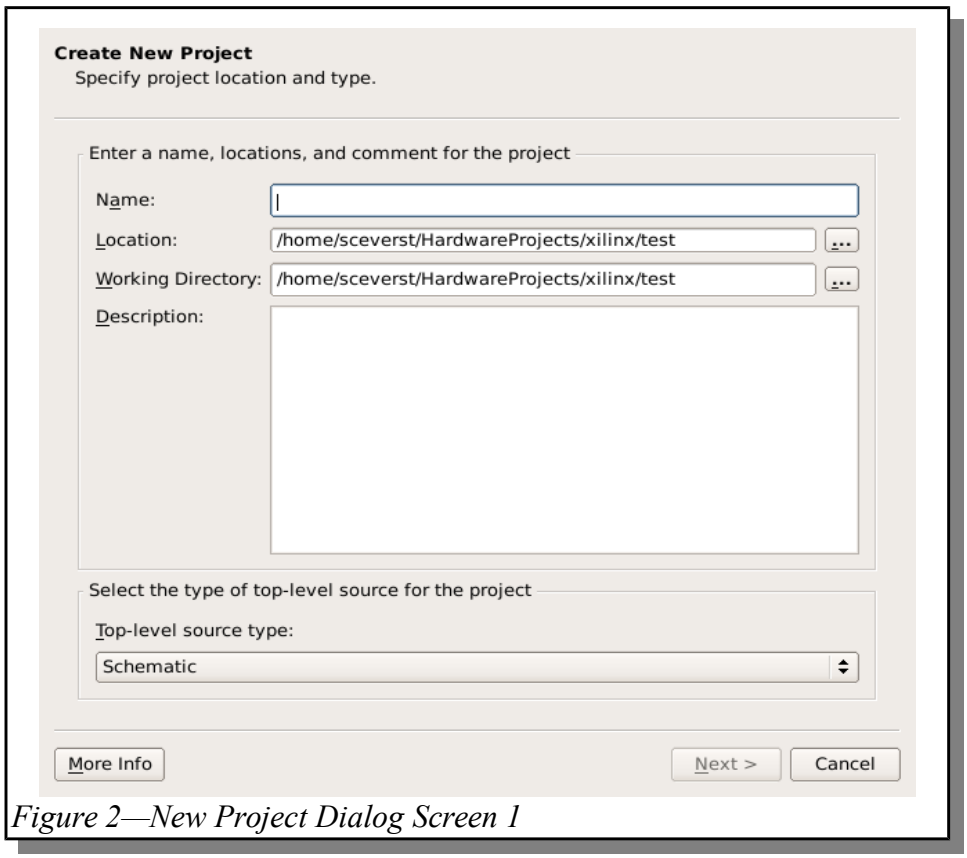


Figure 2—New Project Dialog Screen 1

Provide a name for the project, a path to where you will find the project (the working directory) and any description you wish to include about the project.

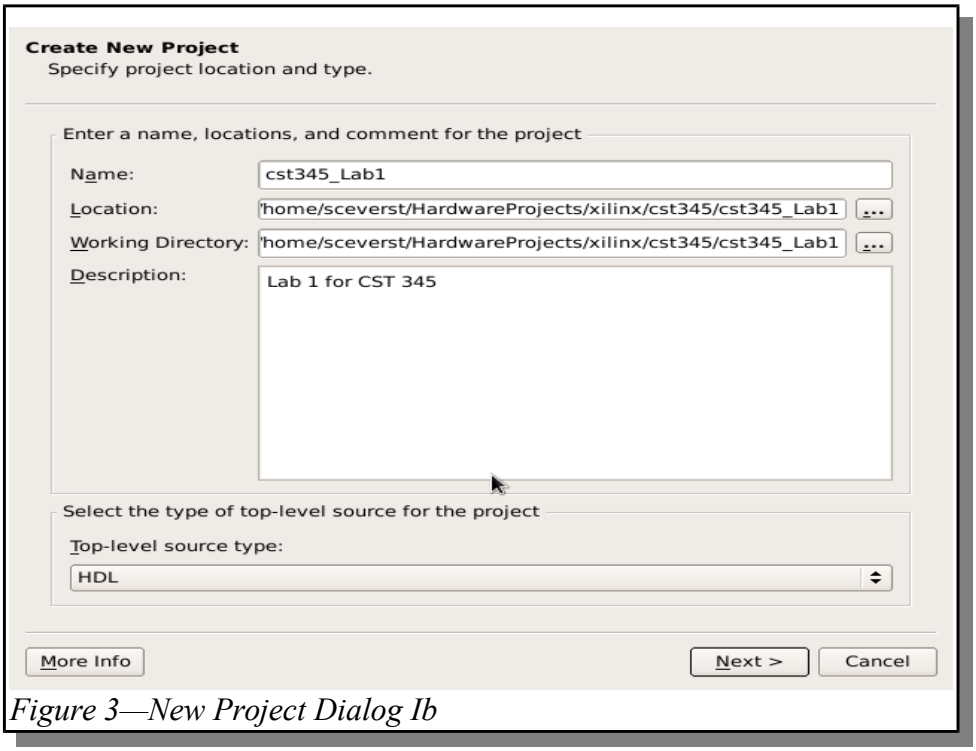


Figure 3—New Project Dialog Ib

You are prompted to enter a project name, a project location, and a top level source type, as shown in Figure 3. You may change the project location to another folder if you wish. Do not use file or folder names that contain spaces. I advise all students to purchase a USB Flash Drive and store their labs on removable media. Never store your projects on the lab machines (with deepfreeze running they will disappear as soon as the system is rebooted). You may also store projects on your Z: drive which is your network home directory. When you are satisfied with the project name and location, click "Next".

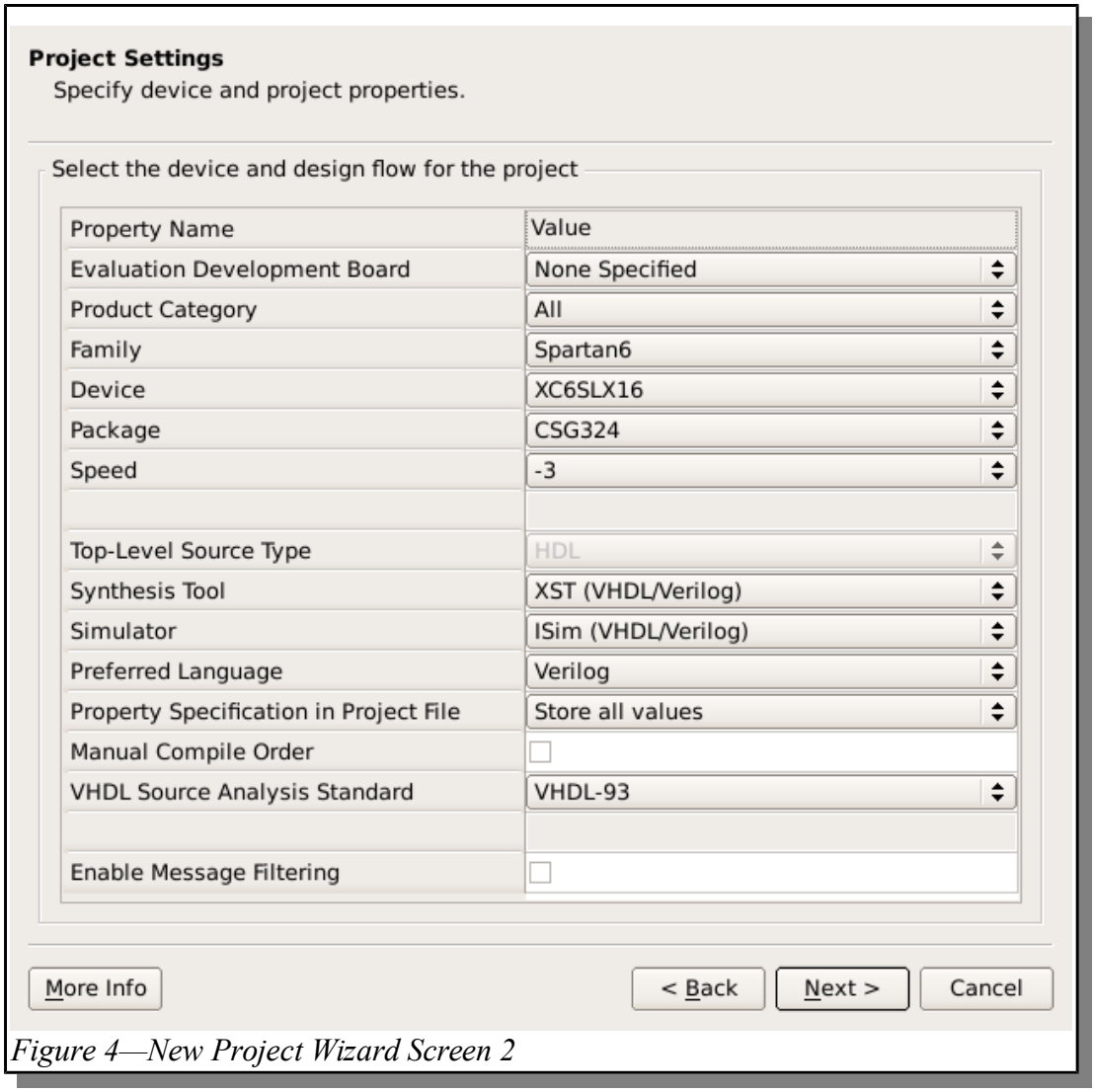


Figure 4—New Project Wizard Screen 2

The next dialog allows you to set additional project options. The first group of settings shown in Figure 4 represents the FPGA that is available to you on the Nexsys 3 board. The second group of settings represents the design entry language, synthesis tool, and simulator tool and preferences. Set options are shown in Figure 4 and click "Next". You will now see a project summary appear on the screen. Click Finish and the ISE design panel appears.

Figure 5 shows the ISE Project Navigator window now appears allowing you to **add or create** new source files to be used in the project.

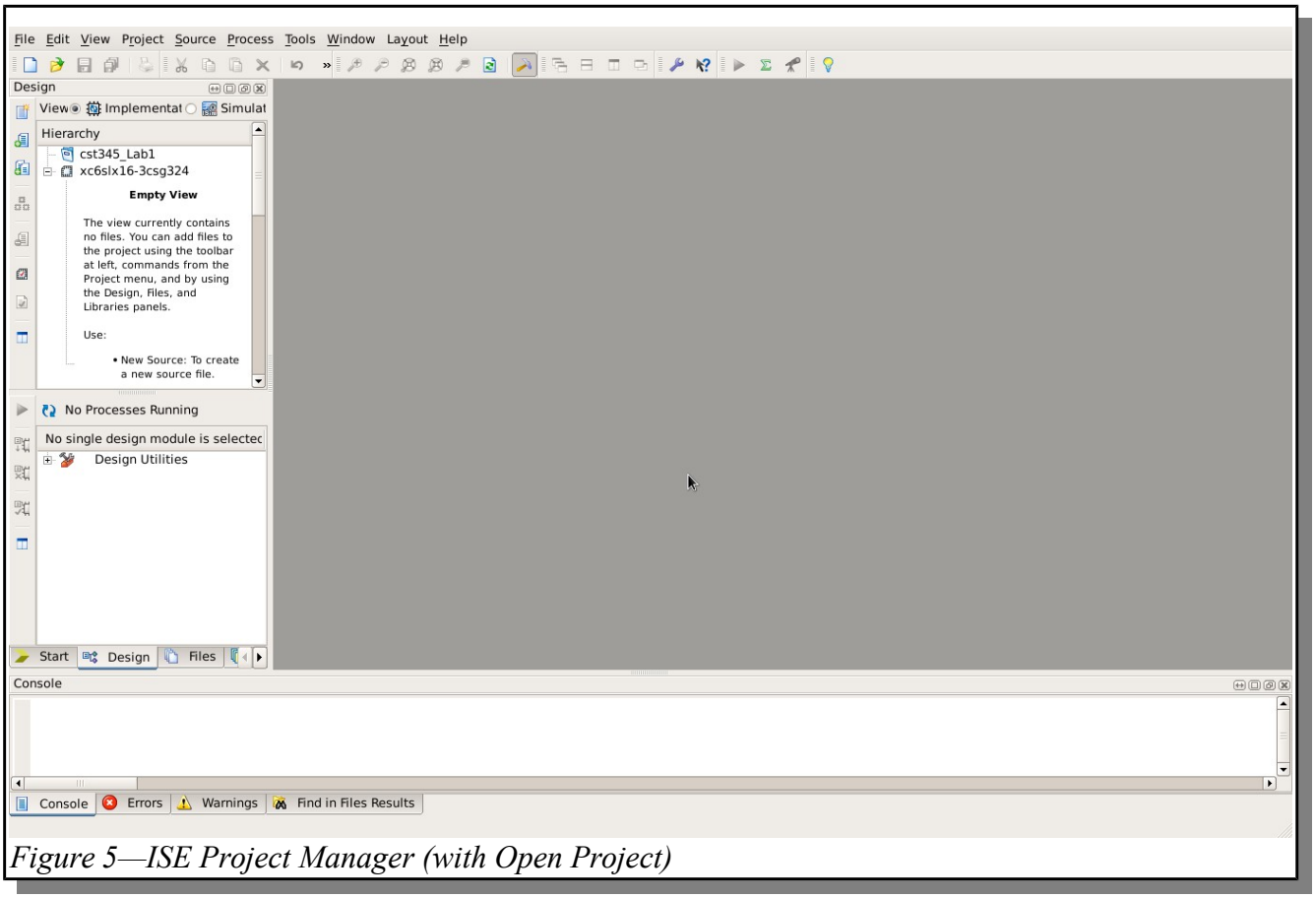


Figure 5—ISE Project Manager (with Open Project)

Creating new source files

If you wish to create a new source file click on the new source ICON on the navigator window. This brings up the new source wizard as shown in Figure 6

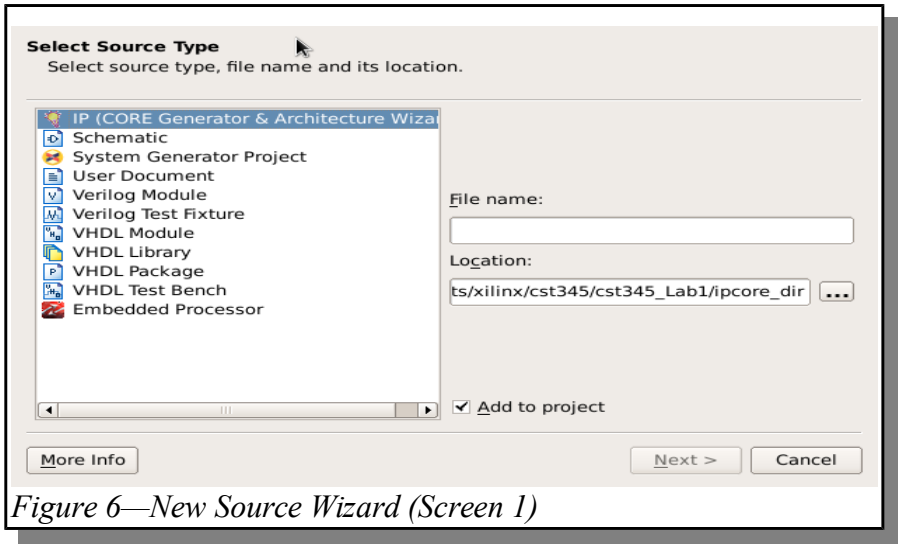


Figure 6—New Source Wizard (Screen 1)

Select *Verilog Module* and give it a filename. Make sure location is correct and that Add to Project is checked as shown in Figure 7. Now click “Next”.

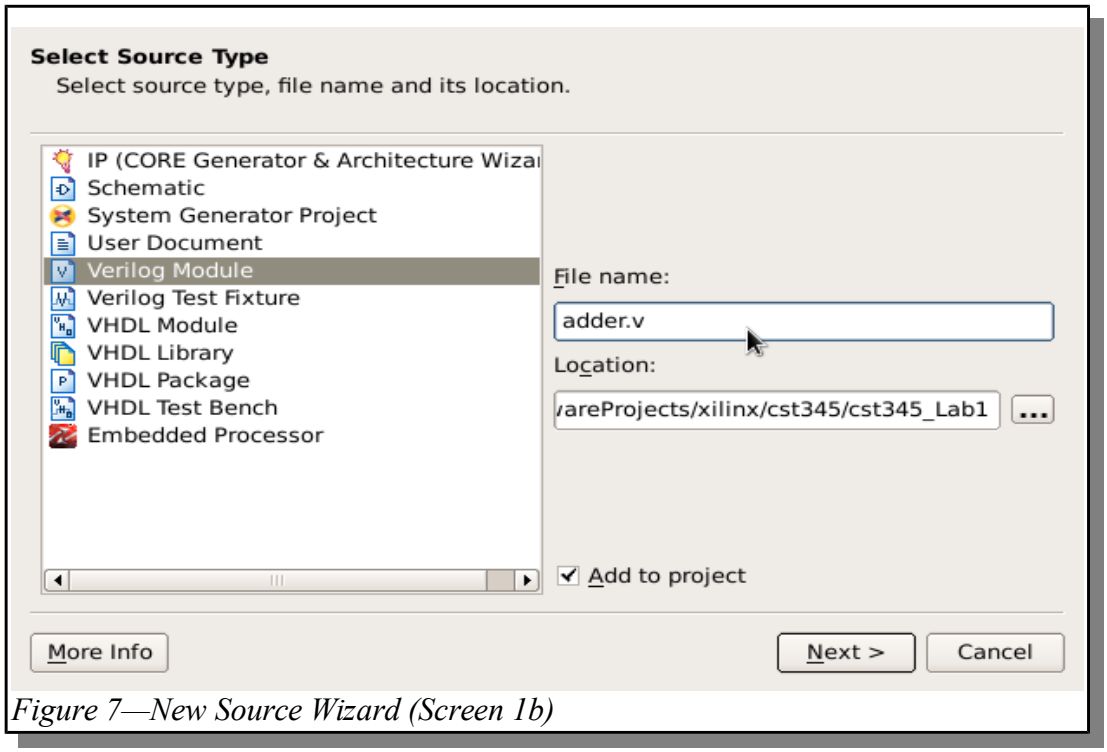


Figure 7—New Source Wizard (Screen 1b)

The next dialog optionally allows you to specify the ports of the module. This may also be done in the text editor, when creating the module, so skip it at this stage. Simply confirm that the settings match those shown in Figure 8 and click “Next”. Then click “Finish”.

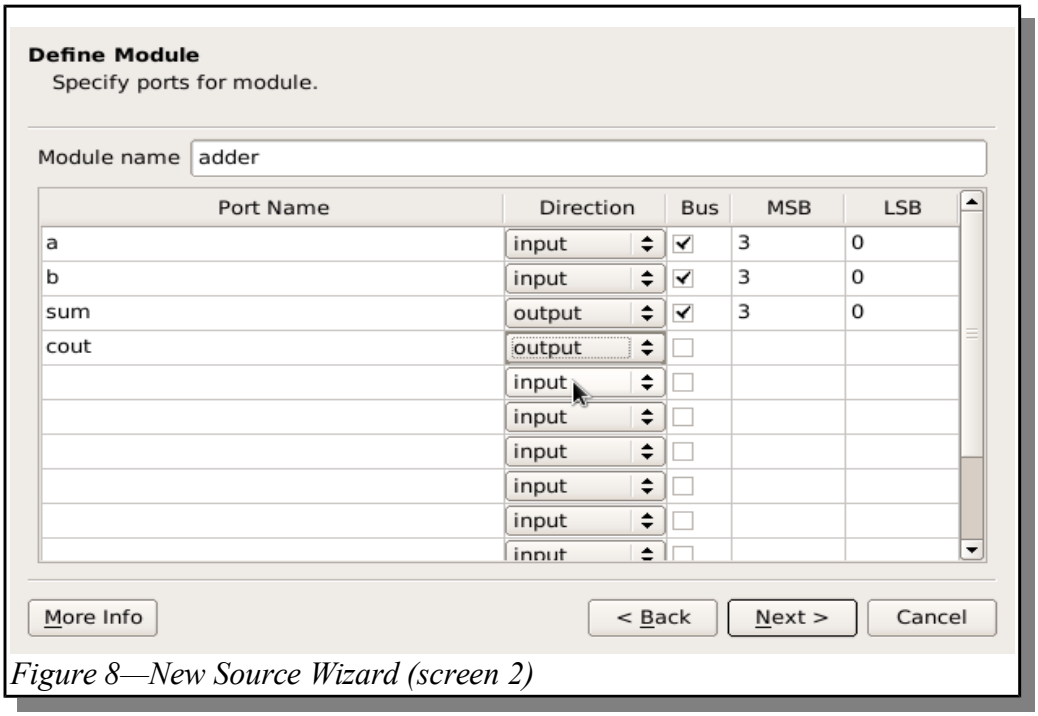


Figure 8—New Source Wizard (screen 2)

The text editor, some of the basic file structure is already in place (although we are going to replace everything that was automatically generated) as shown in Figure 9. Keywords are displayed in blue, data types in red, comments in green, and values in black. This color-coding enhances readability and recognition of typographical errors. Now, replace everything in your adder design file with the code shown in Figure 10.

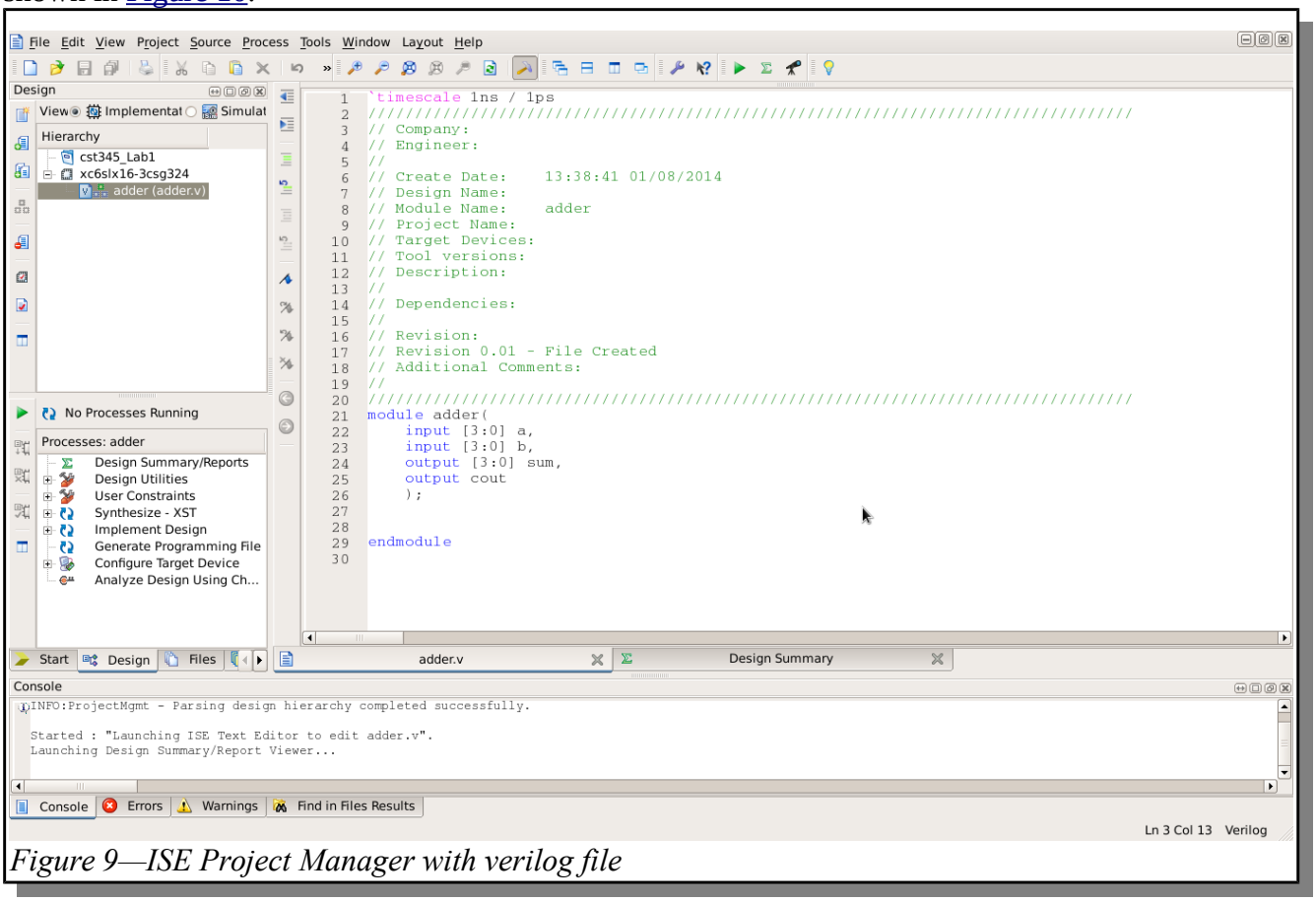
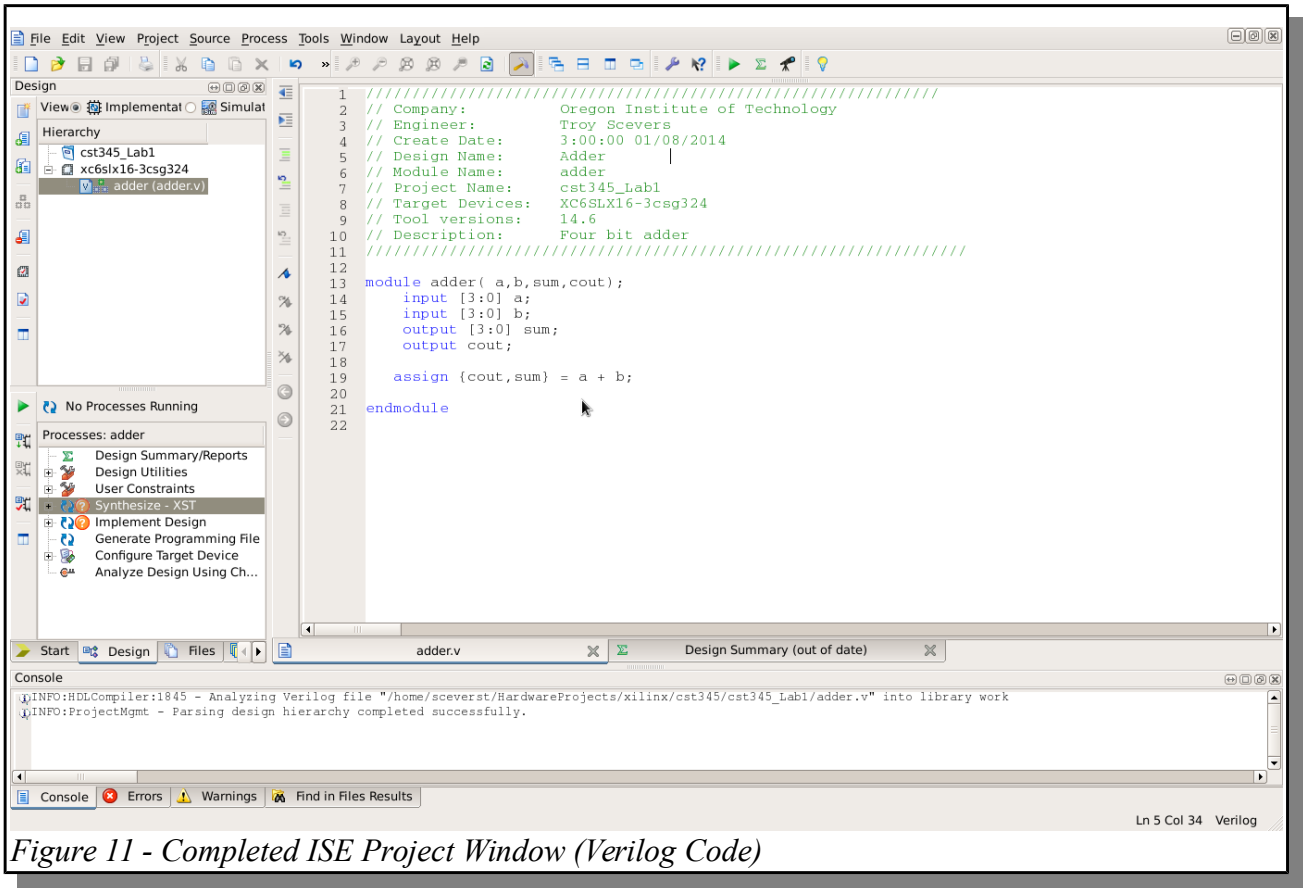


Figure 9—ISE Project Manager with verilog file

```
//////////////////////////////////////////////////////////////////
// Company:              Oregon Institute of Technology
// Engineer:             Troy Scevers
// Create Date:     3:00:00 01/08/2014
// Design Name:          Adder
// Module Name:     adder
// Project Name:         cst345_Lab1
// Target Devices:  XC6SLX16-3csg324
// Tool versions:   14.6
// Description:          Four bit adder
//////////////////////////////////////////////////////////////////

module adder( a,b,sum,cout);
    input [3:0] a;
    input [3:0] b;
    output [3:0] sum;
    output cout;

     assign {cout,sum} = a + b;

endmodule
```

Figure 10 - Verilog Source Code

At this point, you should end up with a window that looks somewhat like that shown in Figure 11 and Figure 12. Once you are satisfied, save the file and close the window. It is a good idea to get in the habit of saving your project. There are options on the main menu to save individual files or the complete project.




```
1  //////////////////////////////////////////////////////////////
2  // Company:          Oregon Institute of Technology
3  // Engineer:         Troy Scevers
4  // Create Date:      3:00:00 01/08/2014
5  // Design Name:      Adder
6  // Module Name:      adder
7  // Project Name:     cst345_Lab1
8  // Target Devices:   XC6SLX16-3csg324
9  // Tool versions:    14.6
10 // Description:      Four bit adder
11 //////////////////////////////////////////////////////////////////
12
13 module adder( a,b,sum,cout);
14     input [3:0] a;
15     input [3:0] b;
16     output [3:0] sum;
17     output cout;
18
19    assign {cout,sum} = a + b;
20
21 endmodule
22
```

Figure 11 - Completed ISE Project Window (Verilog Code)

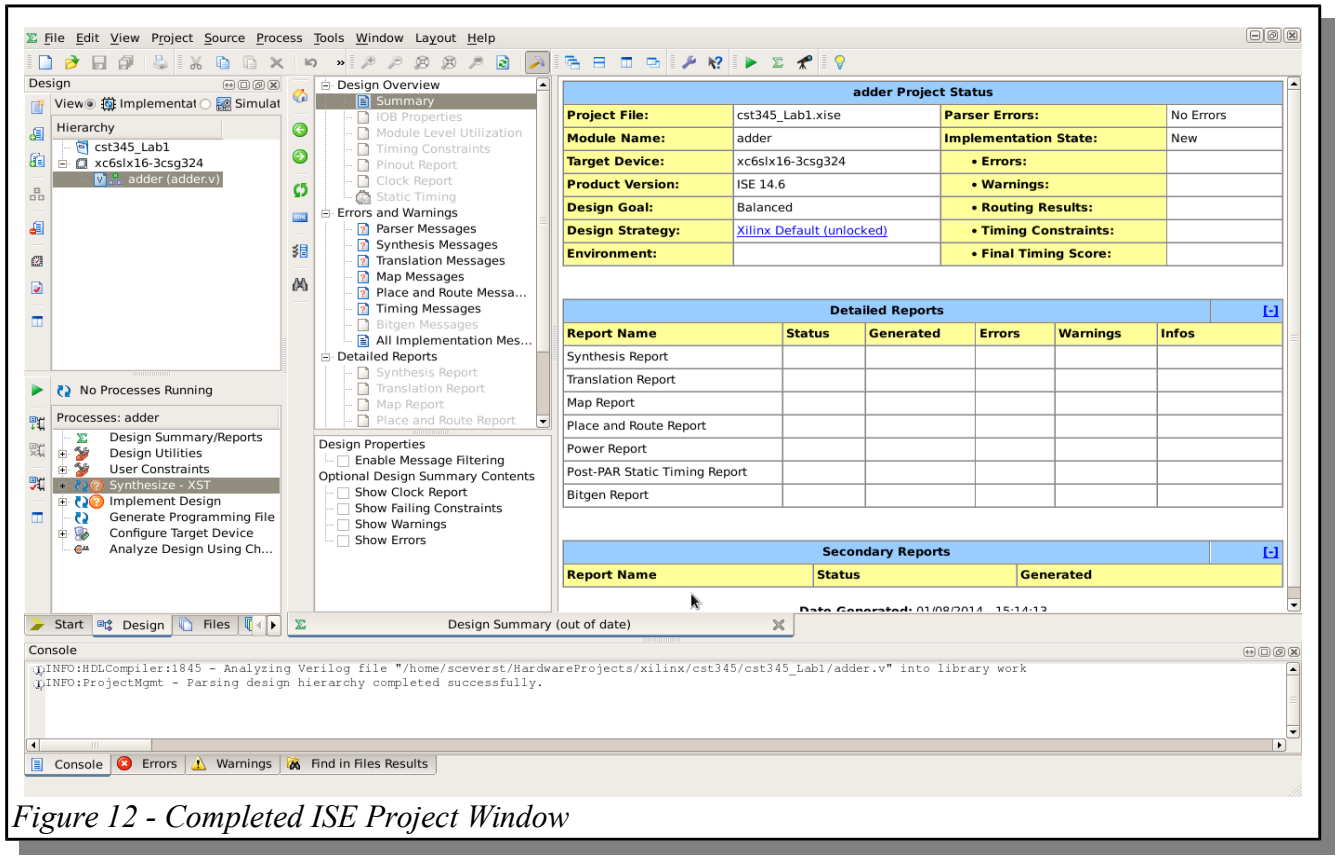


Figure 12 - Completed ISE Project Window

Behavioral Simulation

Functional simulation is done before the design is implemented to verify the logic you have described is correct. This allows a designer to find and fix any bugs in the design before spending time with subsequent steps. Project Navigator provides an integrated flow with the ISE Simulator (ISim) that allows simulations to be run from Project Navigator.

In order to simulate the design, a test bench is required to stimulate the design. Create a new source file for the test bench. Either select Project -> New Source from the main menu or use the equivalent process in the Processes window. The first of the New Source dialog boxes will appear, as shown in Figure 13. You now need to create a Verilog test Fixture for testing the counter.

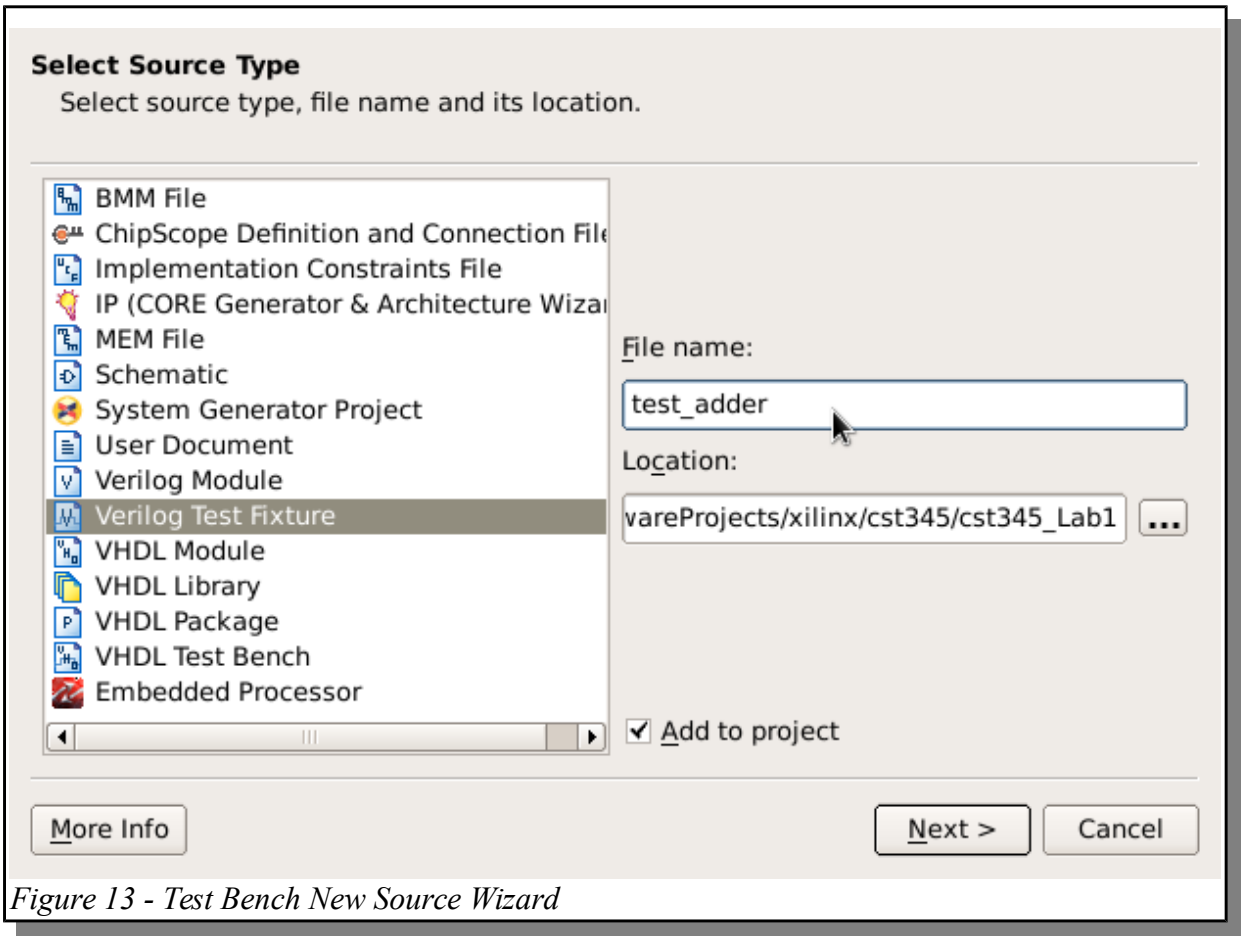


Figure 13 - Test Bench New Source Wizard

Select Verilog Test Fixture to indicate you are creating a Verilog-HDL test bench module. Then, provide a file name as shown in Figure 13. You should not need to change the specified location, which should be inside the project directory you created earlier. Click “Next”.

The second dialog, shown in Figure 14, asks you to identify a design module with which the test bench should be associated. Select the counter design as shown and click “Next”.

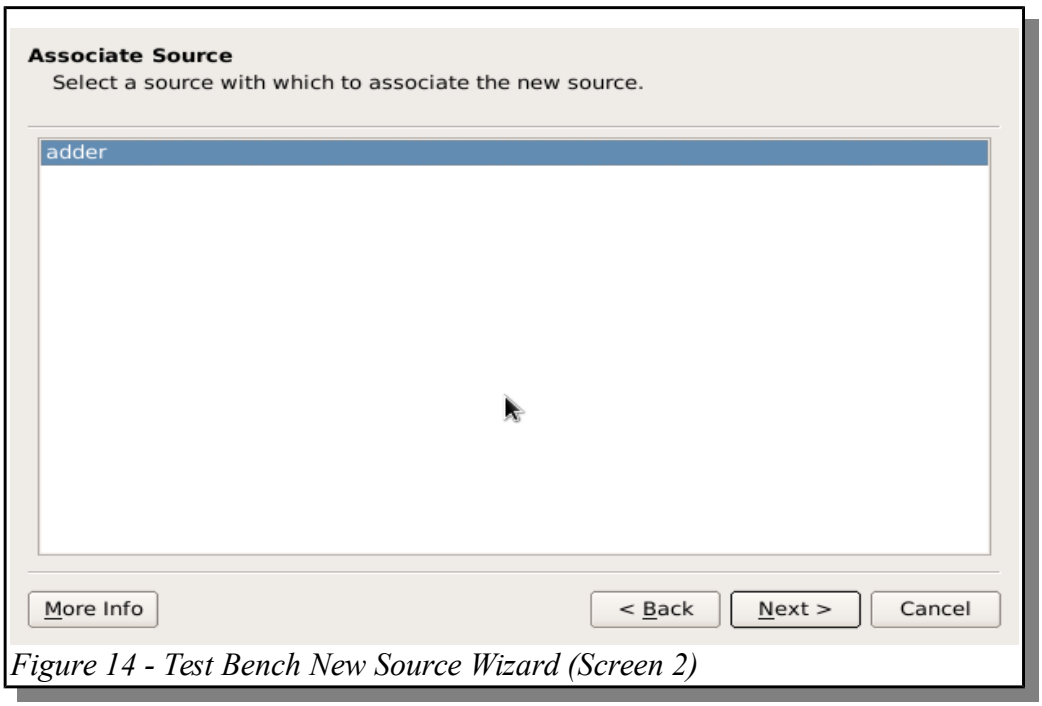


Figure 14 - Test Bench New Source Wizard (Screen 2)

Review the summary to make sure it matches what is shown in Figure 15. Click “Finish” to complete this process. The new source file may automatically open in the text editor, but if it does not—set the Sources window to display items for Behavioral Simulation and then double click on the source file icon in the window.

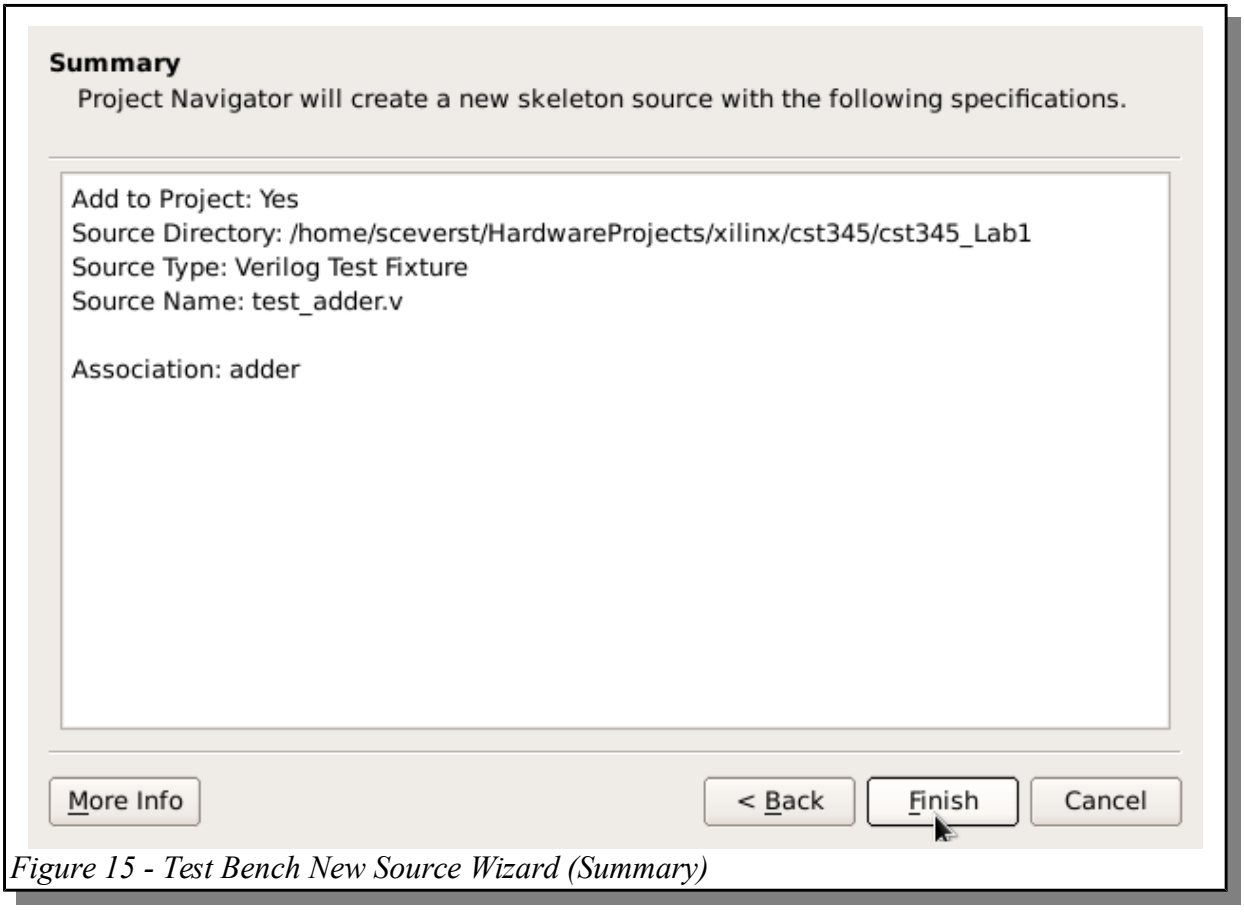


Figure 15 - Test Bench New Source Wizard (Summary)

In the text editor, some of the basic file structure is already in place although we are going to replace everything that was automatically generated. Keywords are displayed in blue, data types in red, comments in green, and values in black. Enter the stimulus code needed for this file and save Now, enter the test bench for the test_adder. You may copy and paste this from Figure 16.

```
`timescale 1ns / 1ps
////////////////////////////////////////////////////////////////////////////////
// Company:         Oregon Institute of Technology
// Engineer:        Troy Scevers
// Create Date:   15:30:09 01/08/2014
// Design Name:   adder
// Module Name:   test_adder.v
// Project Name:  cst345_Lab1
// Verilog Test Fixture created by ISE for module: adder
////////////////////////////////////////////////////////////////////////////////
module test_adder;
          // Inputs
          reg [3:0] a;
          reg [3:0] b;
          // Outputs
          wire [3:0] sum;
          wire cout;
          // Instantiate the Unit Under Test (UUT)
          adder uut (
                    .a(a),
                    .b(b),
                    .sum(sum),
                    .cout(cout)
          );
          reg test_passed_add;
           reg test_passed_cout;
          initial begin
                    // Initialize Inputs
                    a = 4'b0000;
                    b = 4'b0000;
                    // Wait 100 ns for global reset to finish
                    #100;
                    $display("At time %t, a = %a, b = %b, sum = %sum, cout = %cout.", $time, a, b, sum, cout);
                    if (sum != 4'b0000) test_passed_add = 1'b0;
                    if (cout != 1'b0) test_passed_cout = 1'b0;
                    // Test Case #1
                    a = 4'b1010;
                    b = 4'b0101;
                    #100;
                    $display("At time %t, a = %a, b = %b, sum = %sum, cout = %cout.",$time, a, b, sum, cout);
                    if (sum != 4'b1111) test_passed_add = 1'b0;
                    if (cout != 1'b0) test_passed_cout = 1'b0;
                    // Test Case #2
                    a = 4'b1111;
                    b = 4'b1111;
                    #100;
                    $display("At time %t, a = %a, b = %b, sum = %sum, cout = %cout.", $time, a, b, sum, cout);
                    if (sum != 4'b1110) test_passed_add = 1'b0;
                    if (cout != 1'b1) test_passed_cout = 1'b0;
          end
endmodule
```

Figure 16 - Test Bench Verilog Code

At this point, you should end up with a window that looks somewhat like that shown in Figure 16. Once you are satisfied with the stimulus you created, save the file and close the window. It is a good idea to get in the habit of saving your project. There are options on the main menu to save individual files or the complete project.

Now that you have a test bench in your project, you can perform functional simulation on your design.

To locate the simulator process, change the button from implementation to the simulation button at the top of the design window. You will see the simulation hierarchy such that the test bench is the top and the Unit under test (uut) is the lower level as seen in Figure 17.

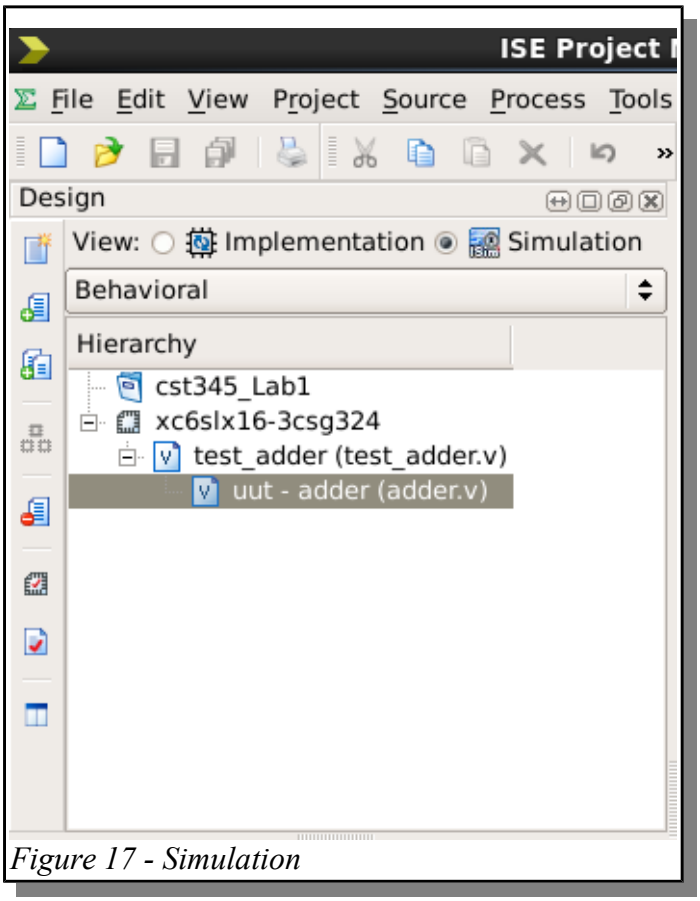


Figure 17 - Simulation

To locate the simulator process, set the Sources window to display sources for Behavioral Simulation. The Xilinx ISim Simulator as an available process in the Source window. Expand the Xilinx ISim Simulator process to reveal Behavioral Simulation as it appears in Figure 18.

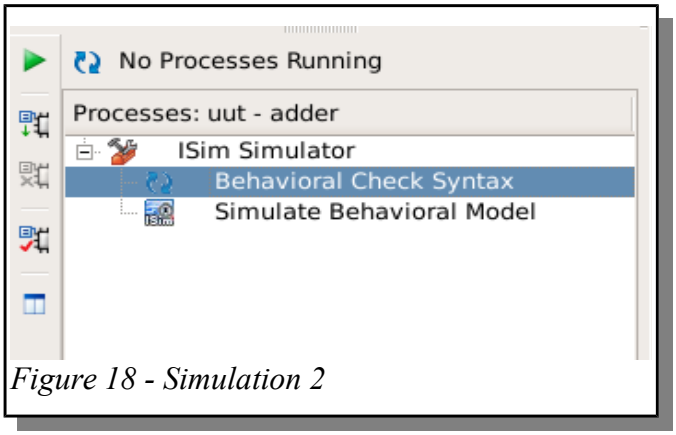


Figure 18 - Simulation 2

To start the simulation, double-click Simulate Behavioral Model. The simulator compiles the source files and runs the simulation. Output is displayed in the console window as well as in a waveform display, as shown in Figure 19.

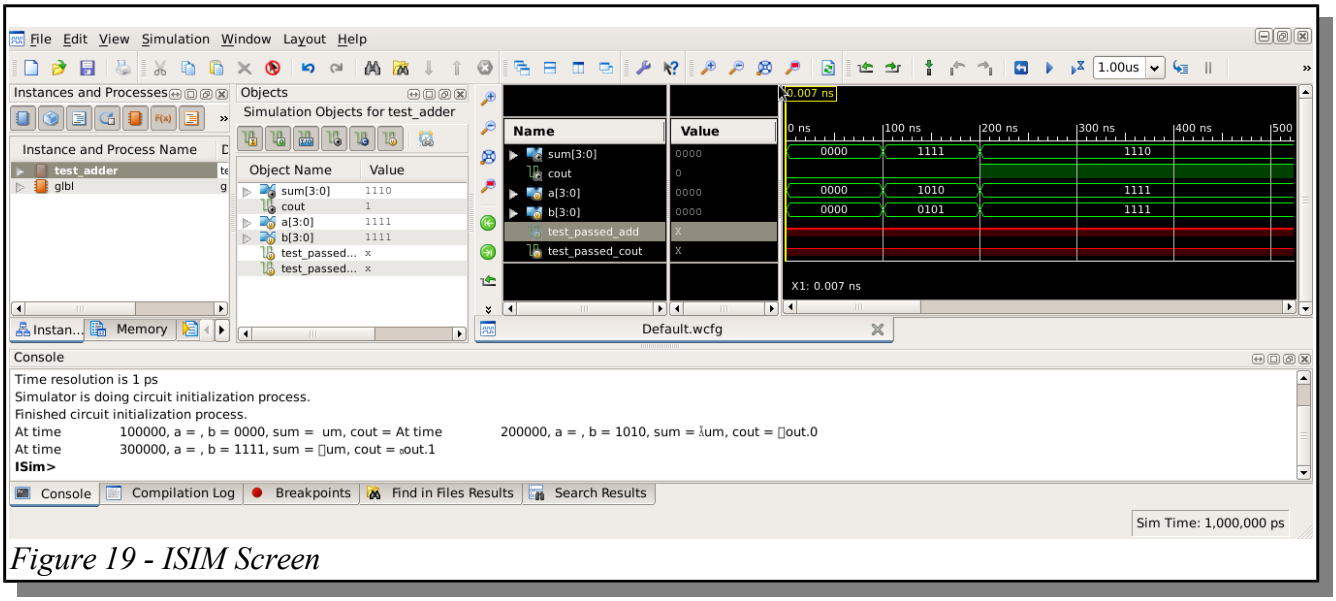


Figure 19 - ISIM Screen

An In-Depth Tutorial form Xilinx (UG682) can be found on the SharePoint site.

PlanAhead Tool

Switch back to Implementation from Simulation (See Figure 17). Now we need to open the PlanAhead tool to assign our inputs and outputs for the board. Go to Tools->PlanAhead->I/O Pin Planning (Pre Synthesis). Once there Assign the pins as shown in Figure 20. You can see more details about where things are assigned on the Nexys3 board by looking at the file Nexys3_rm.pdf on the SharePoint Site.

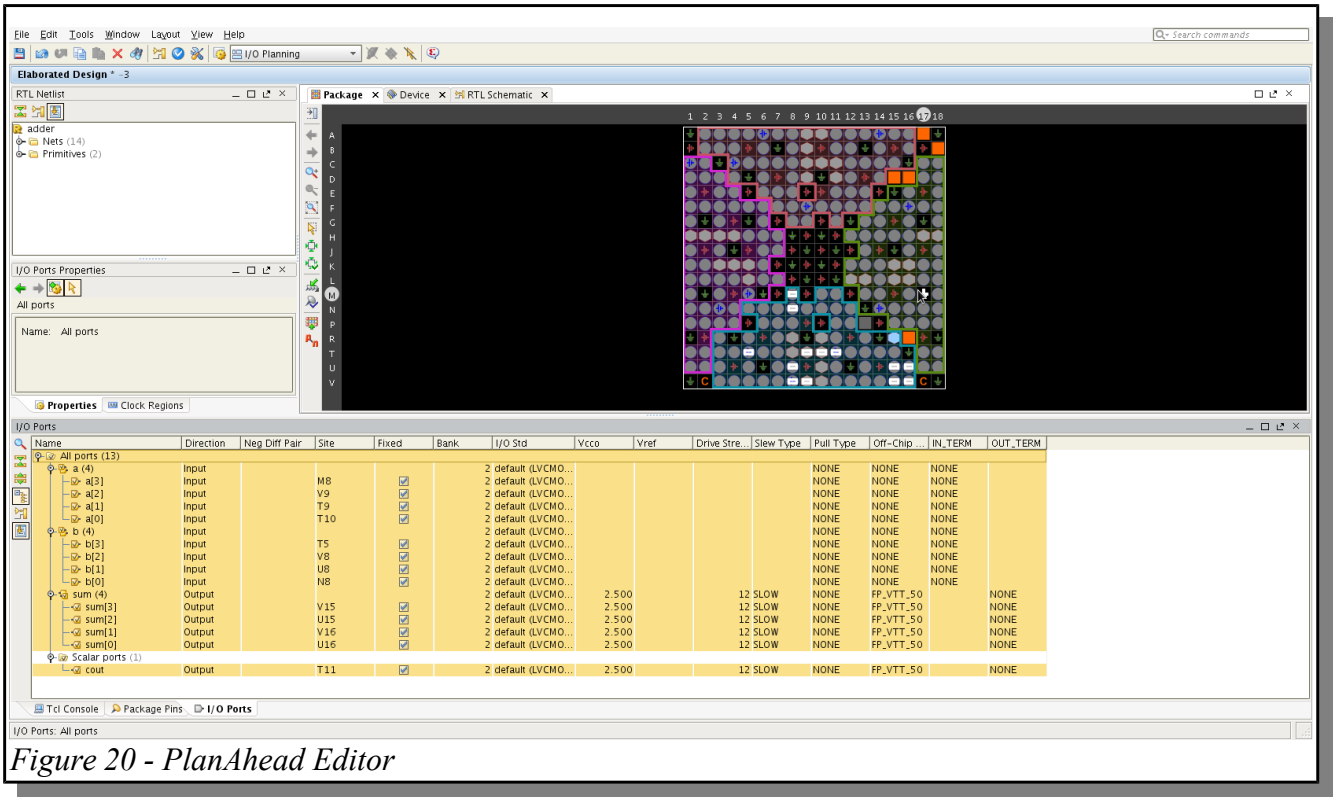


Figure 20 - PlanAhead Editor

Now it is time to Implement our Top Level. Click on the Green Arrow icon to do so. This will synthesize the code, Translate it, Map it to our technology and finally perform a Place & Route. Next make sure your board is plugged in and double click on the “Configure Target Device”. This will bring up the impact tool as shown in Figure 21. Configure your device and Demo to lab instructor for checkoff.

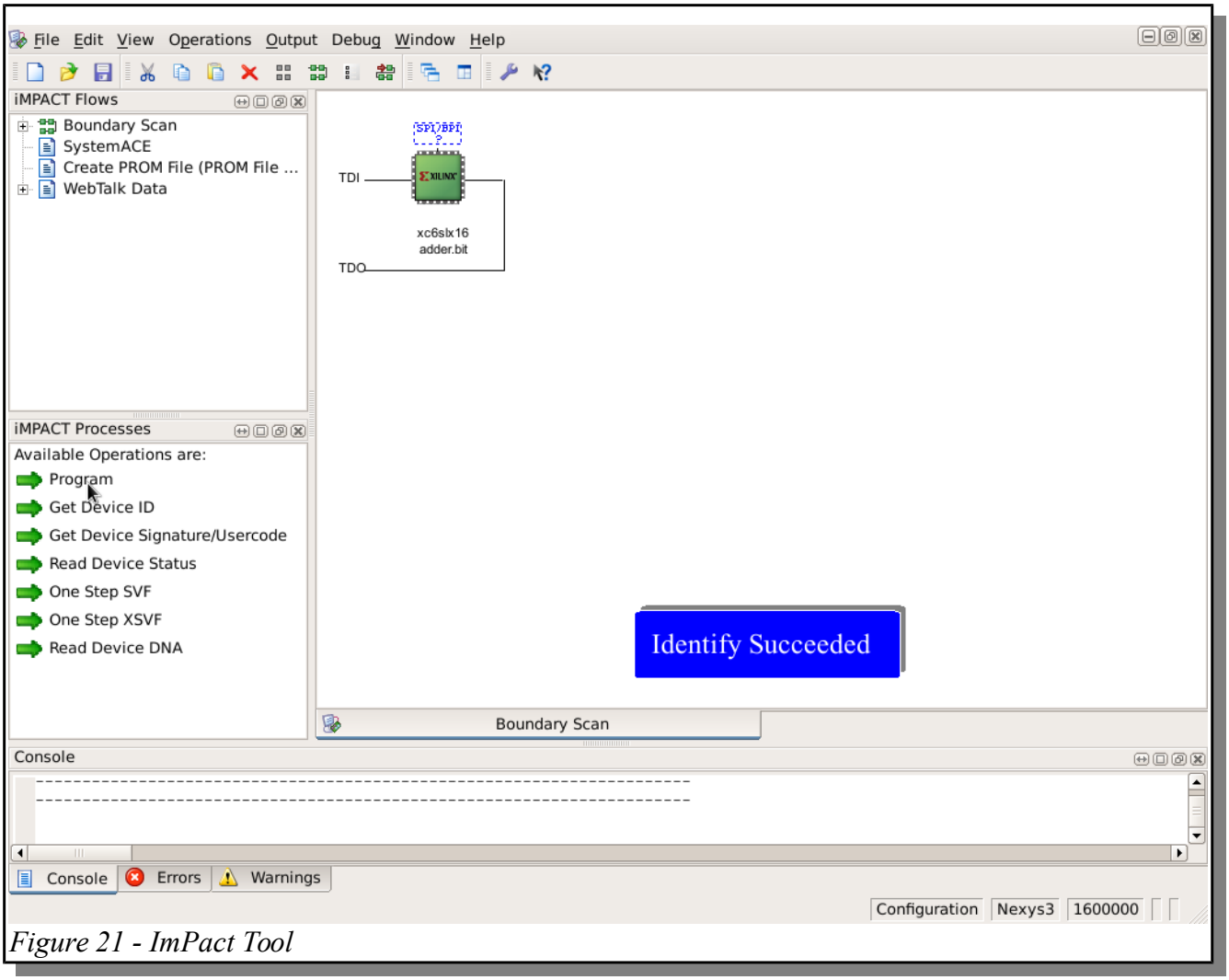


Figure 21 - ImPact Tool