

SANTA CLARA UNIVERSITY
Electrical Engineering Department

ELEN 127 Advanced Logic Design (2019)

Xilinx Vivado Tutorial

Adapted from ELEN 33 material developed by Andy Wolfe

In this tutorial, you will learn how to make use of the Vivado software package to capture the details of a design and download it onto a Nexys 4 FPGA board.

The Vivado Program is called Vivado 2019.1, and it can be found by clicking on the <applications> folder on the Desktop, then the <EE> folder, then <Xilinx Design Tools> folder, and finally it will be in the <Vivado 2019.1> folder. Click the program called Vivado 2019.1, NOT the Tcl shell, command prompt or HLS applications that are also in the folder.

Creating a New Project

1. To create a new project, select the *Create Project* item from the startup window.

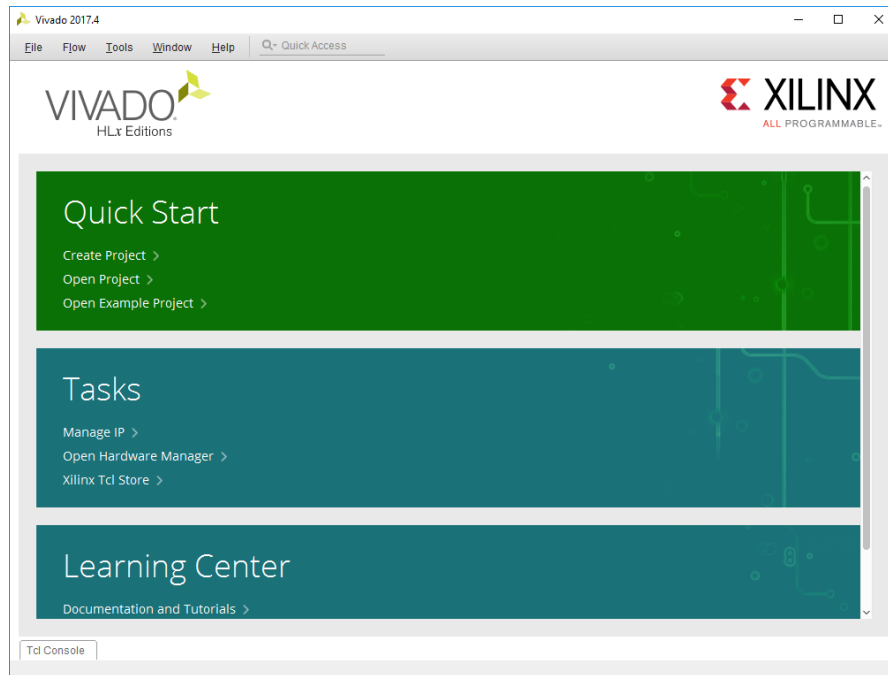


Figure 1: Create New Vivado Project Window

2. Click *Next* to move to the next window.

3. Make sure to create a folder on the Z drive to store your Lab. Saving it on the local desktop could prevent Vivado from properly synthesizing the design. Vivado has some strange rules about path names in some versions including no spaces in the path name. We recommend creating a project subdirectory for each project. Click *Next* to continue.

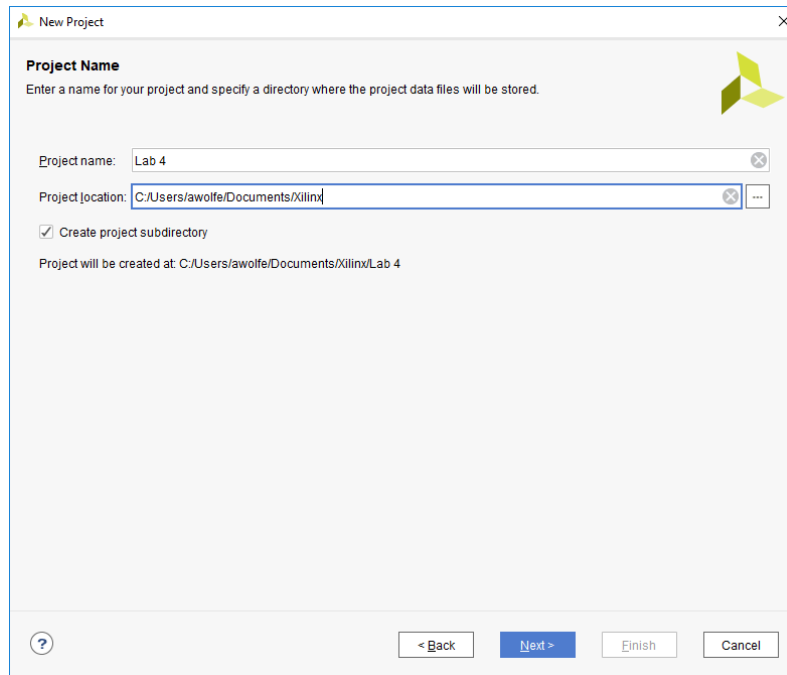


Figure 2: Project Name Window

4. Ensure *RTL Project* and *Do not specify sources at this time* are checked and click *Next* to proceed.

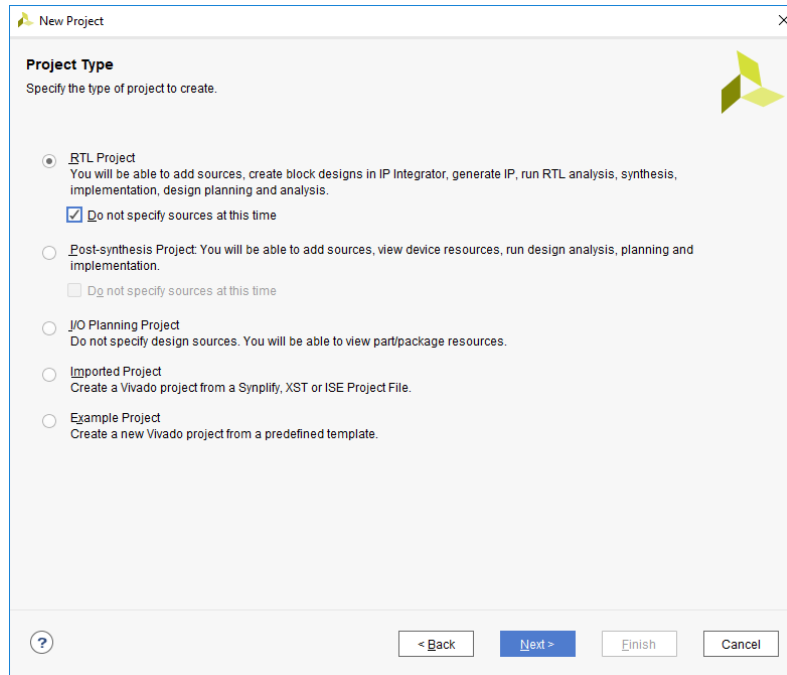


Figure 3: Project Type Window

5. Choose the Artix-7 part with a csg324 package. The full part name is xc7a100tcsg324-2. Click *Next* to continue.

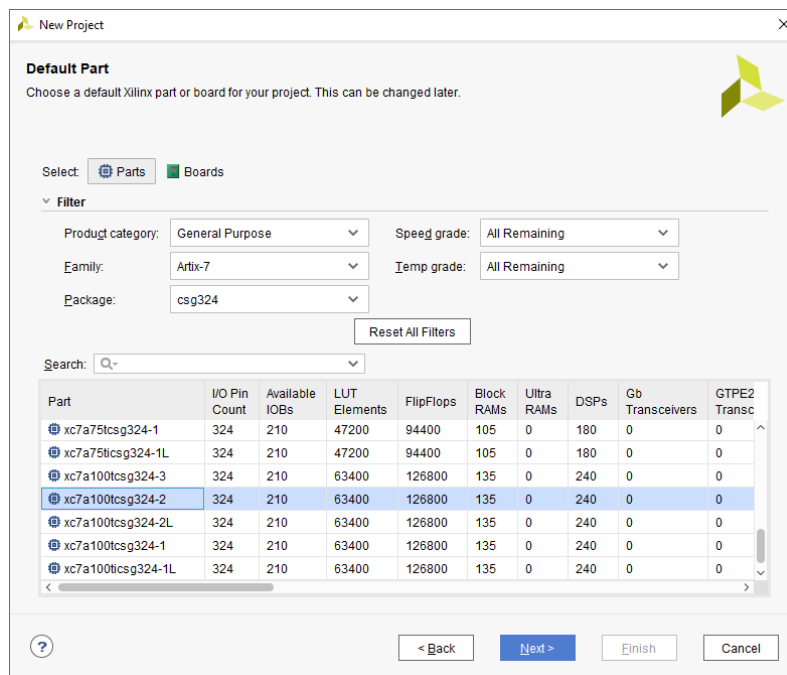


Figure 4: Default Part Window

- Click *Finish* to create the project. You should now see the Project Summary tab open in the main window:

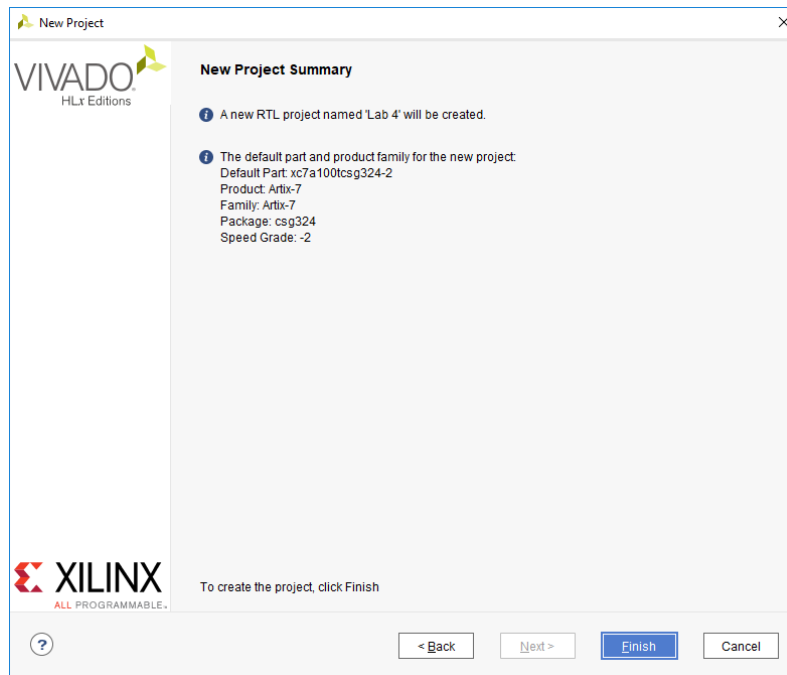


Figure 5: New Project Summary Window

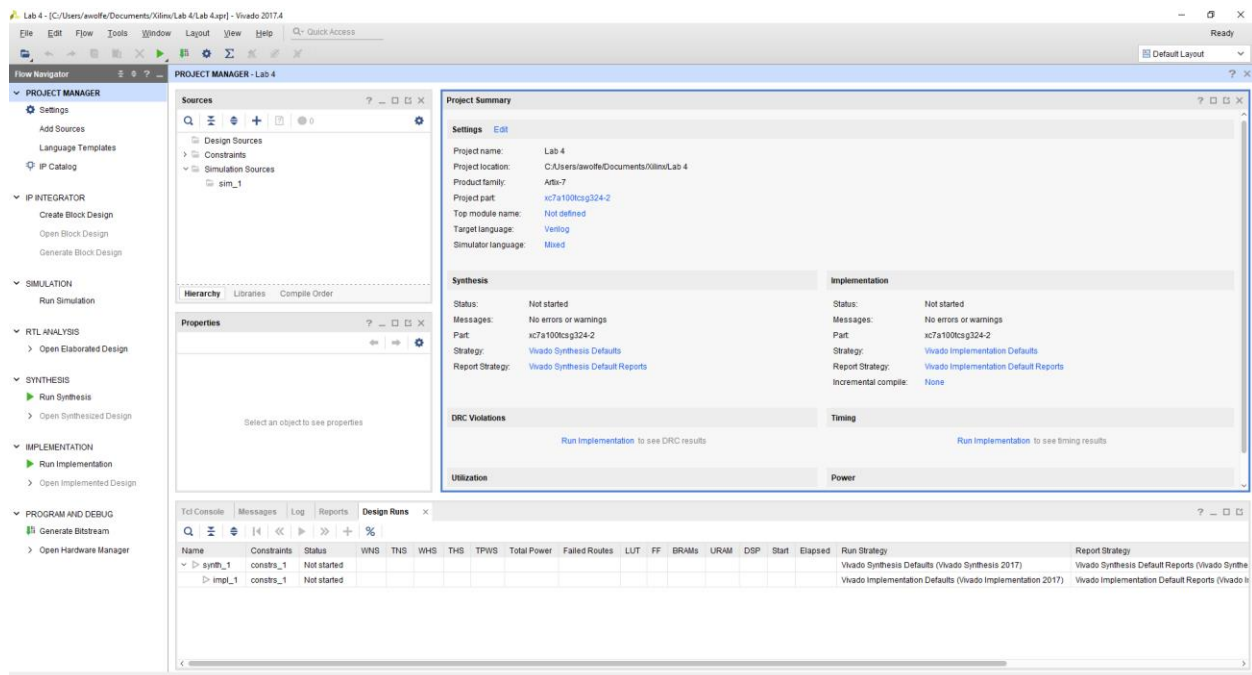


Figure 6: Project Summary Window

- Click *Create Block Design* selection in the left window. It is under the *IP Integrator* category.

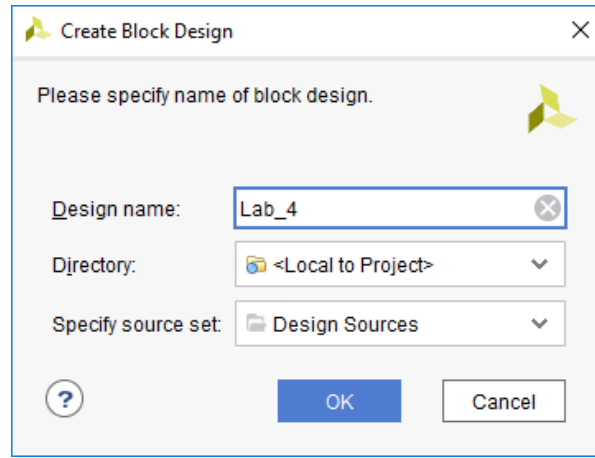
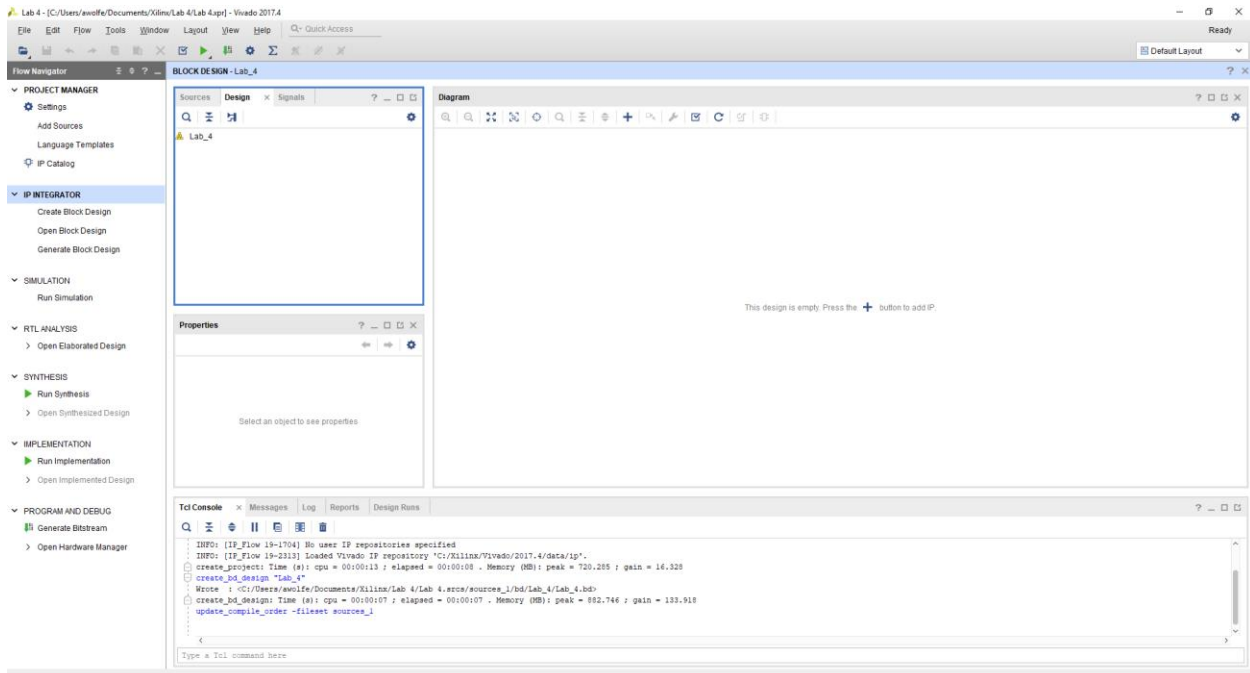


Figure 7: Create Block Design Window

- Choose a name for the project and click *OK*. The design tab should now replace the project summary tab in the main window.



Adding an Adder-Subtractor IP Core to Your Project

This portion of the tutorial shows you how to add what is essentially a library component. In this case, an adder/subtractor. While we will likely not use library components like this, many of the steps are generally applicable.

1. In the middle of the Design window, the following message should be present:

This design is empty. Press the **+** button to add IP.

2. Click on the symbol and the following window should appear:

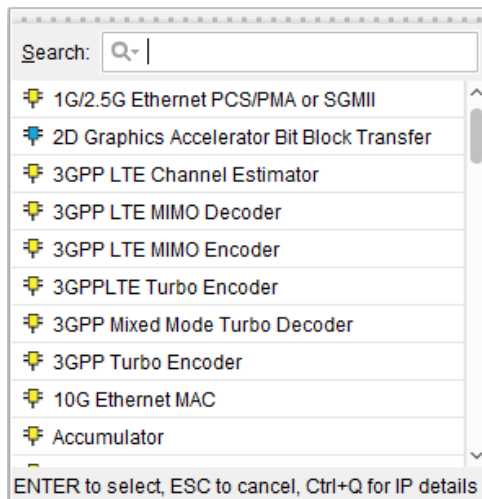
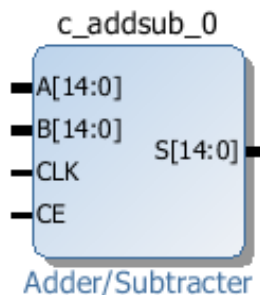


Figure 8: IP Search Window

3. Select the Adder/Subtractor from the selection and hit *Enter*. The following block should now appear in the design window:



4. Double click on the middle of the block to re-customize the adder/subtractor¹ to our specifications. Be careful to select the entire object and not one of the words on it. The following window should appear:

¹ I really think this word is spelled "subtractor" but apparently Xilinx does not.

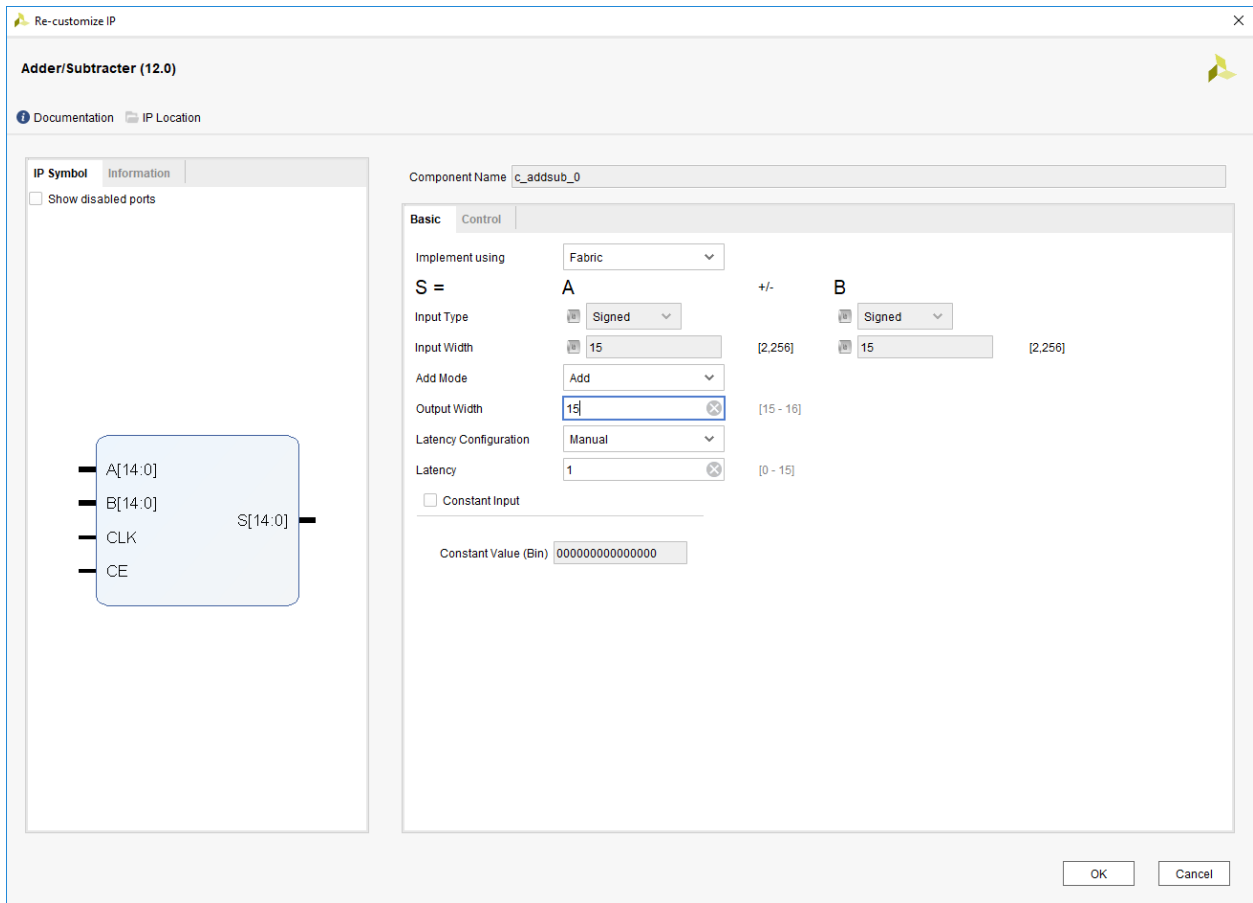
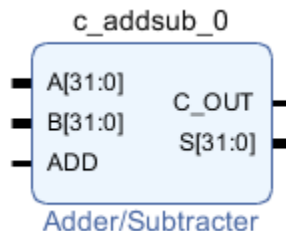


Figure 9: Adder/Subtractor Re-customize Window

Experiment with the “Show Disabled Ports” option to see the turned-off features.

5. Change the A and B bus width to 32 bits in the input width boxes. The box next to each input needs to be clicked for each one in order to change the bus width from Auto to Manual. Change each input width to 32 bits. Change the output width to 32. Also, change the latency to 0, the add mode to “Add Subtract” and both input types to unsigned. The Control tab at the top of the window should be clicked next. Finally, uncheck the clock enable checkbox and check the Carry Out box then click *OK*. The modified adder block should now look as follows:



Next, you will learn how to simulate this design.

Verifying and Simulating the Adder-Subtractor

1. Right click an empty space in the design window and select Create Port from the pop-up menu:

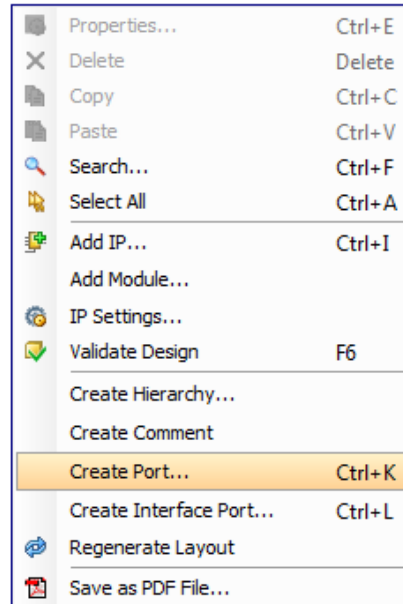


Figure 10: Design Window Pop-up Menu

2. In the new window, assign the port name as “a”. Make sure the direction is set to input. Check the *Create Vector* checkbox and make sure the range is from 31 to 0. Finally, click OK. The *Create Port* window looks as follows:

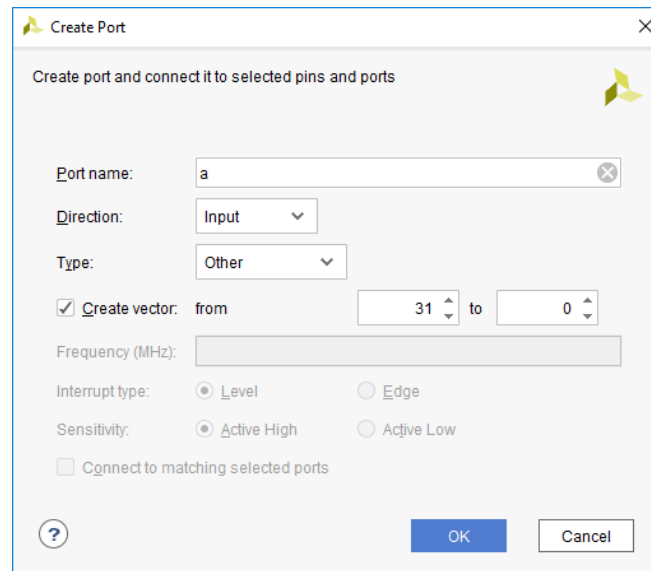


Figure 11: The Create Port Window

3. Create two more input ports. The two port names should be “b” and “add”. The “b” port should be a vector from 31 to 0 just like the “a” port. The “add” port should not be a vector.
4. Create two output ports called “s” and “c_out”. The “s” port should be a vector that goes from 31 to 0. The “c_out” port is not a vector.

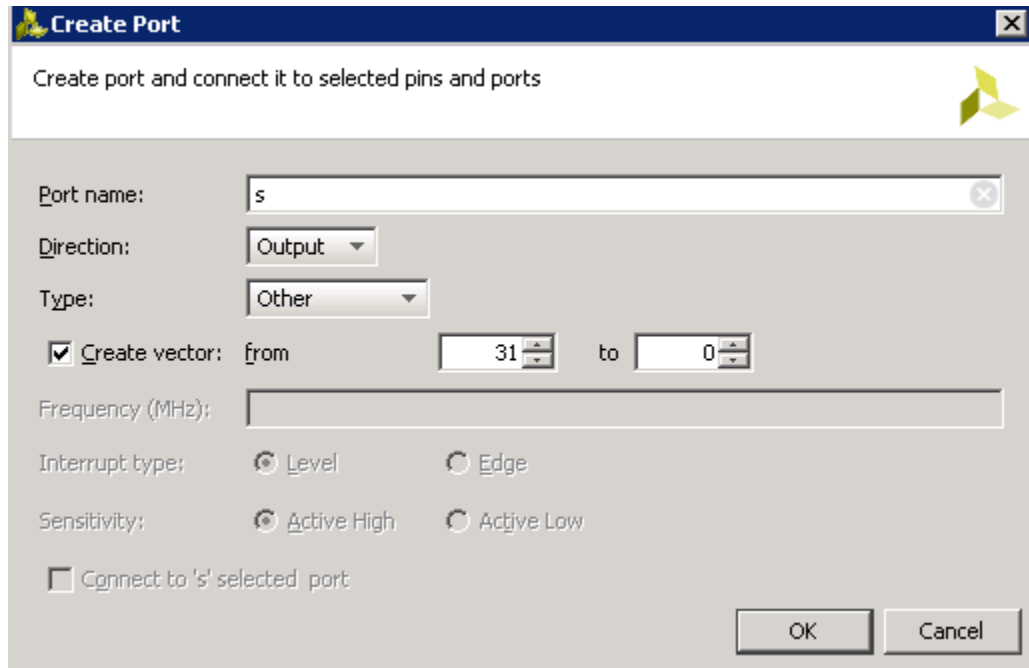


Figure 12: s Port Configuration

You should now have five ports in the design window that look similar to the following:

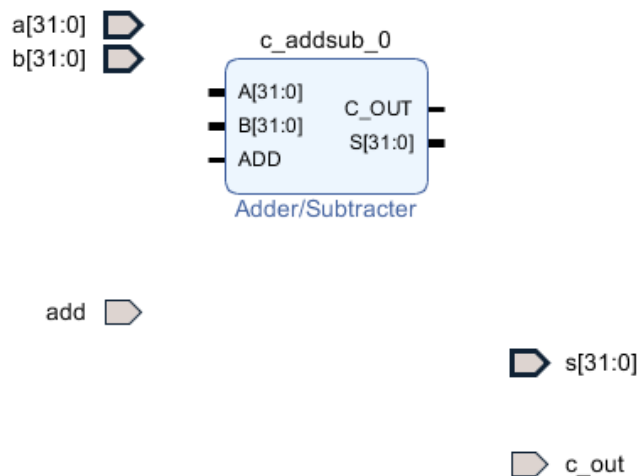


Figure 13: Adder/Subtractor Block With Unconnected Ports

- Next, place the mouse over the tip of one of the ports and ensure it turns into a pencil icon. Click and hold the left mouse button to draw a line from the port to the adder/subtractor. Connect the ports to their corresponding labels on the adder/subtractor. The end result should look something like the following:

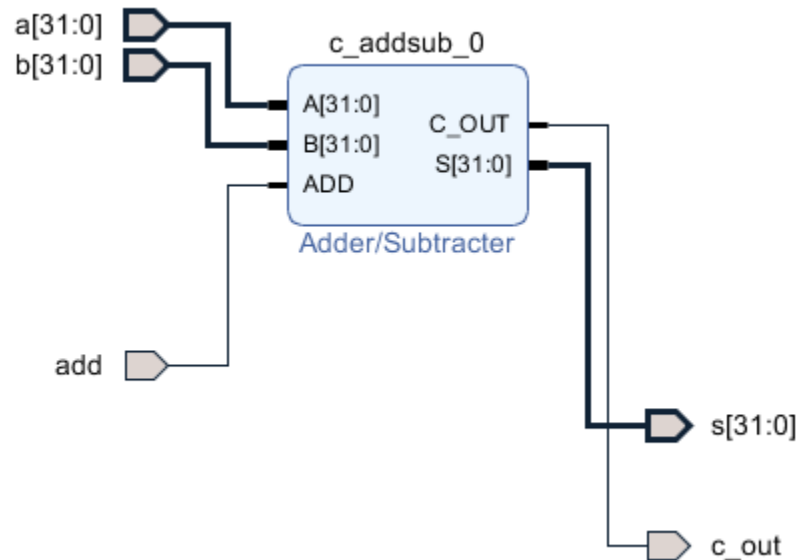


Figure 14: Module with Connected Ports

- The block diagram can be cleaned up by right clicking on an empty part of the design and selecting Regenerate Layout. The end result should look like this:

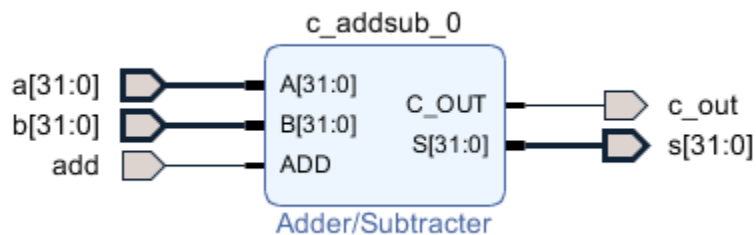


Figure 15: Regenerated Design

7. You will create a verilog-based testbench to test your design. Pick the Sources tab. Right click in the *Sources* window and select *Add Sources* from the menu. The following window will appear. *Make sure the Add or Create Simulation Sources* item is selected. Click *Next*.

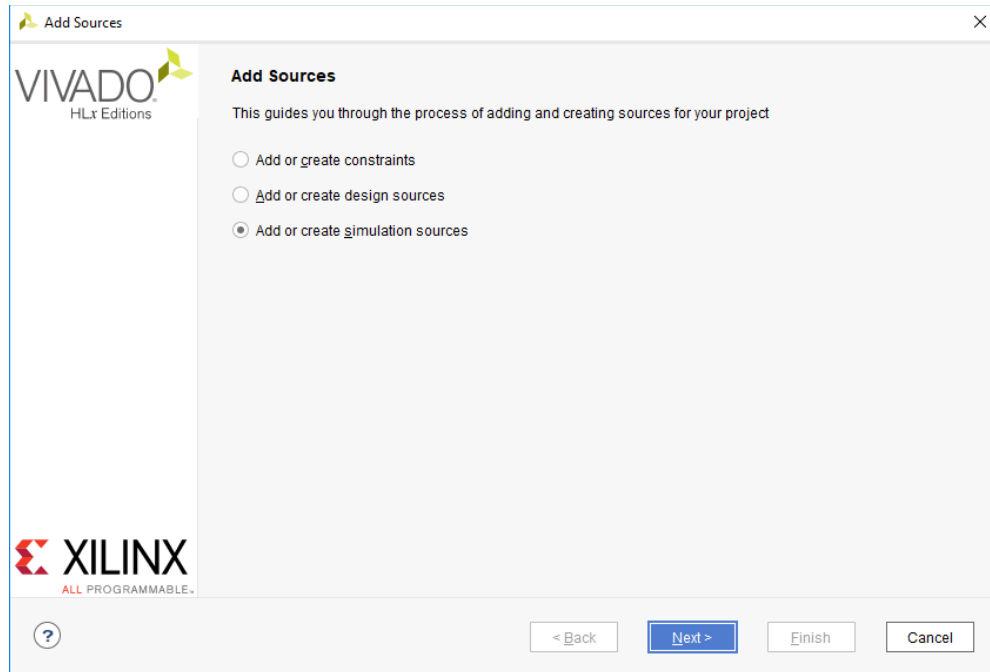


Figure 16: Add Sources Window

8. On the next window, click *Create File* as seen below:

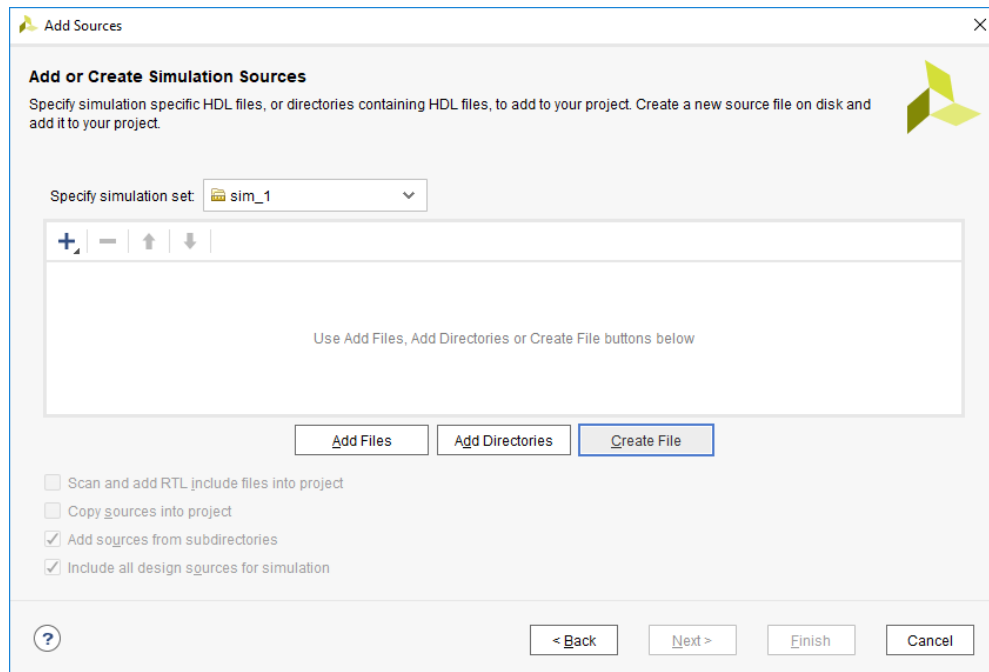


Figure 17: Create New Testbench Window

9. In the next window that appears, name your file `adder_tb` and click *OK*:

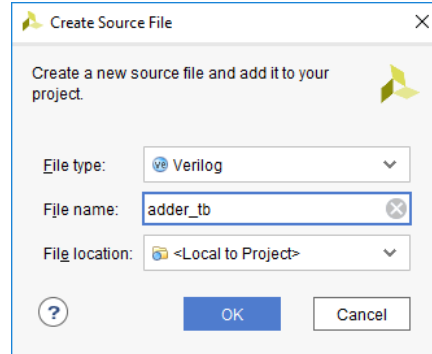
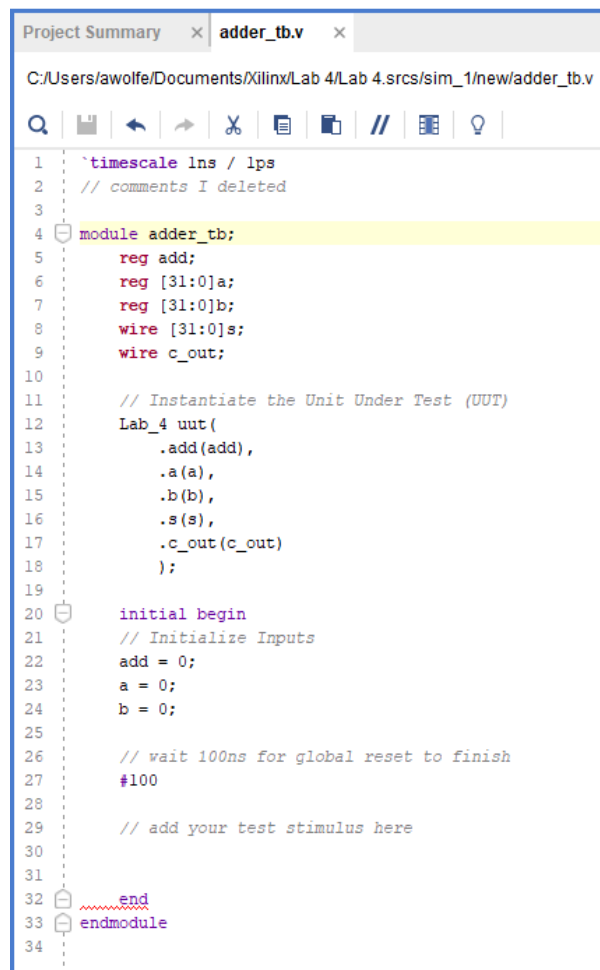


Figure 18: Create Source Window

10. Click *Finish* on the Add Sources window. Click OK and Yes to be done creating the testbench file. (If you get a define module window – just click ok)
11. Double click on the `adder_tb` file that appears in the left hand window to open and edit it. Add the commands shown below. (You need to delete some ())

A screenshot of a code editor window showing a Verilog testbench file named 'adder_tb.v'. The editor has a toolbar with icons for search, save, undo, redo, cut, copy, paste, and other standard editing functions. The code is as follows:

```
1  `timescale 1ns / 1ps
2  // comments I deleted
3
4  module adder_tb;
5      reg add;
6      reg [31:0]a;
7      reg [31:0]b;
8      wire [31:0]s;
9      wire c_out;
10
11     // Instantiate the Unit Under Test (UUT)
12     Lab_4 uut(
13         .add(add),
14         .a(a),
15         .b(b),
16         .s(s),
17         .c_out(c_out)
18     );
19
20     initial begin
21         // Initialize Inputs
22         add = 0;
23         a = 0;
24         b = 0;
25
26         // wait 100ns for global reset to finish
27         #100
28
29         // add your test stimulus here
30
31
32     end
33 endmodule
34
```

Figure 19: The Adder Testbench

You do not need to know Verilog or even understand all the details in this file. To complete the testbench, all you need to understand is how to specify the test vectors, that is, the inputs a , b , and add , to the adder-subtractor. Suppose that you would like to determine the output when the inputs are as follows: $a = 10$, $b = 20$, and $add = 1$. You will have to insert the following statements in the area pointed to by the arrow in the preceding figure:

```
a = 10;  
b = 20;  
add = 1;
```

Note that each statement is terminated by a semi-colon. The following statement represents a delay of 100 time units:
#100

The mode can be changed from add to subtract by changing the input signal add . If you wish subtract the inputs after another 200 time units, you can add the following statements as well to the testbench:

```
#200  
a = 200;  
b = 100;  
add = 0;
```

Add any other test vectors to the testbench and save it. It is important to note that these test vectors should be placed within an initial block (that is, between the keywords *initial begin* and *end*) as shown. After this, you can run the simulation.

Also note that the object that creates the unit under test (uut) has the same name as your block diagram file. If the name is different, the design will not simulate.

Once the design is ready for testing, Click the *Run Simulation* item in the left window. Select Run Behavioral Simulation from the pop up menu. If there are no errors in the design, the simulation will run and the result will be as follows:

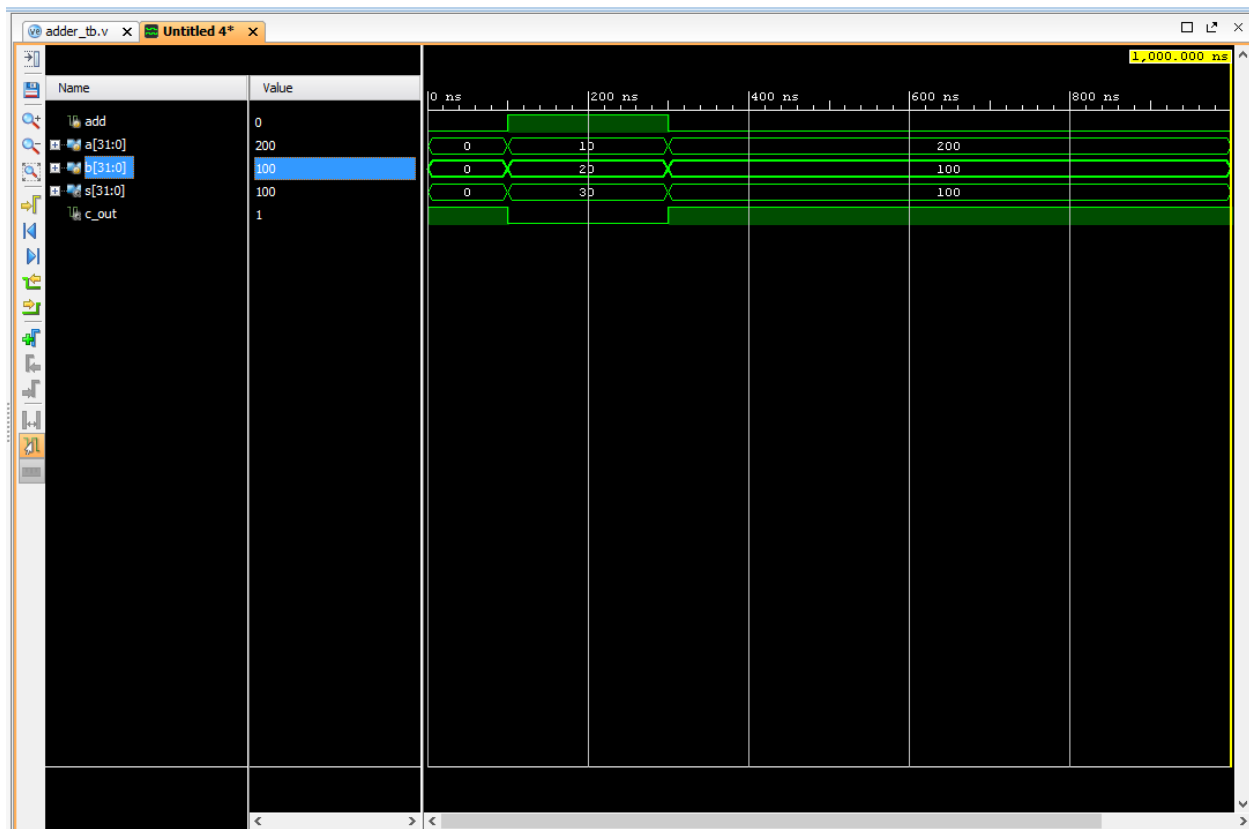


Figure 20: Simulation Results

Experiment with the simulator and figure out how to zoom in and out on the waveforms. Also, figure out how to change from the default number format to signed decimal to make the results easier to read. Create enough simulation tests that you are confident that your circuit works as expected.

Programming the Nexys 4 Board

Mapping the I/O of the processor to the Nexys 4 board

Vivado uses a .xdc file to map signals in your design to pins of the FPGA, so that you can do things like use the switches to provide inputs, or generate outputs to control the 7-segment displays. Your TA will provide an xdc file for you to use in your first lab.

Assuming the file were called “Nexys4_Lab7.xdc”, and that you have copied it to your project directory, add the “Nexys4_Lab7.xdc” file to your project with the “Add or create constraints” option from the “Add Sources” window.

You should now see the constraints file added to the project:

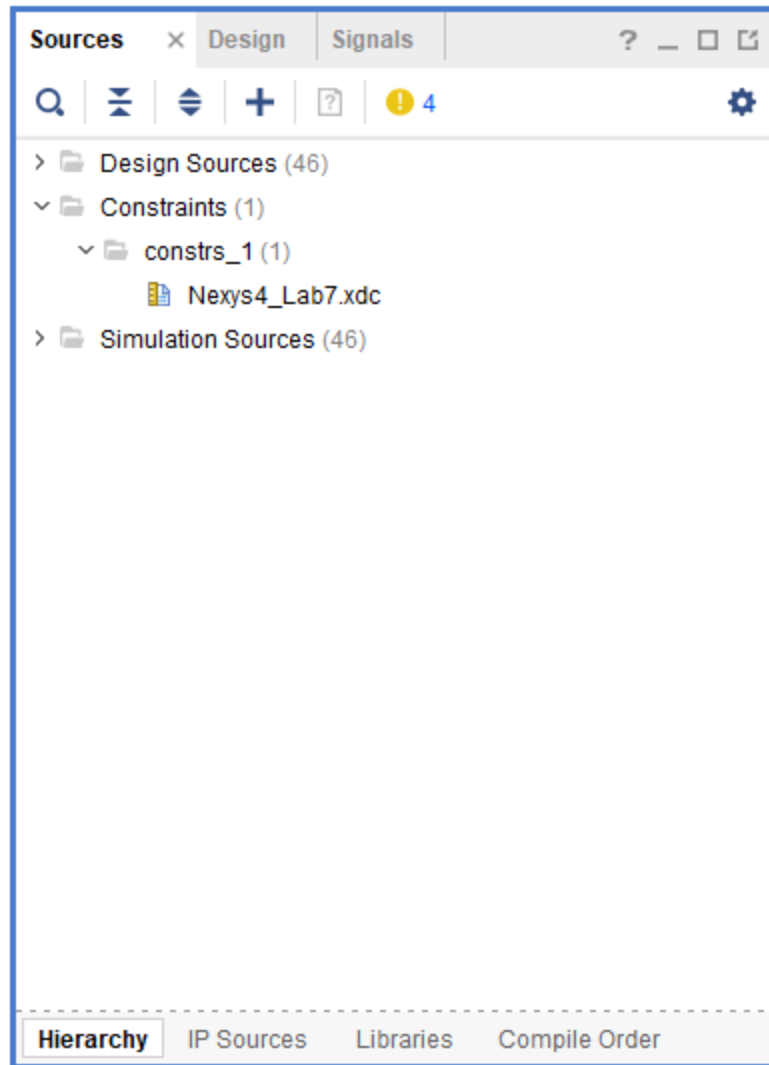


Figure 14: Constraints File Added to the Project

Creating a design wrapper

Open the Design Sources item in the Sources menu and right click on the block diagram file (.bd file) to open the menu. Select “Create HDL Wrapper.” Choose “Let Vivado manage wrapper and auto-update” then OK. It then takes a minute or two to update the design menu.

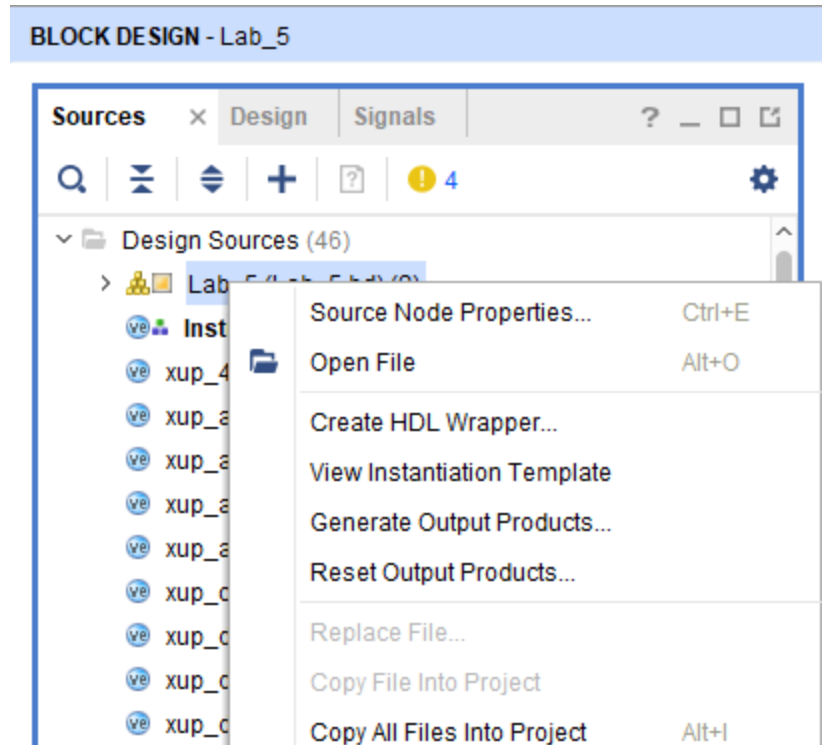


Figure 3: Creating an HDL wrapper

Programming the Nexys4 board

Once the program is ready to be tested, click on the “Generate Bitstream” text in the left-hand window:

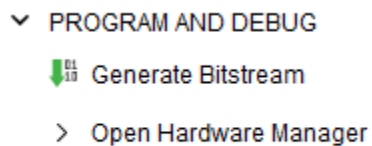


Figure 4: Generate Bitstream

Vivado will now synthesize the design and make it ready for the board. This may take several minutes (or even more than that). Once Vivado has completed writing the bitstream, connect a Nexys 4 board to the computer with the USB cable and turn the power switch on. load the bitstream into the board by selecting “Open Hardware Manager” ->“Open Target” -> “Auto Connect” from the same window. You may get a prompt to open the Hardware Manager.

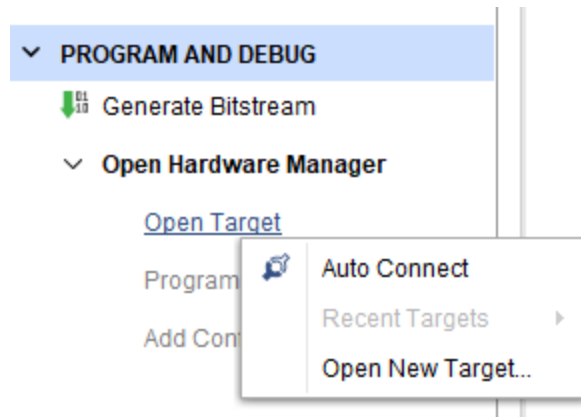


Figure 5: Connecting to the Nexys 4 Board

After the software has connected to the board, a bar at the top of the main window should appear:

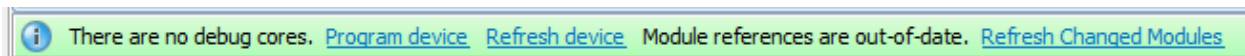


Figure 6: Program Device Bar

Click the “Program Device” link and select the only option available. Another window will pop up. Click the “Program” button on the pop-up window. The board should be ready to use once this is done.