

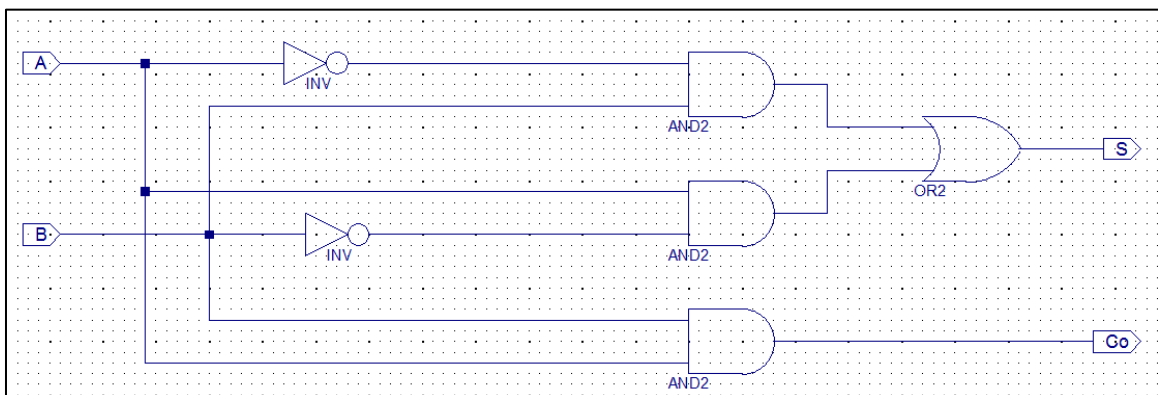
Xilinx ISE 10.1 Tutorial (Building a Half Adder)

Sections

Introduction.....	1
General Xilinx Tips.....	2
Creating a Project.....	3
Schematic Entry	5
Behavioral Simulations (Testing)	8
Implementation	11
Timing Simulation	13
Programming the Board.....	14
Snapshots and Archives	17

Introduction

In this tutorial, you will build a simple half adder, as shown below.



The following pages have screenshots that will guide you through the design, testing, and implementation phases. Use this as a reference throughout the semester.

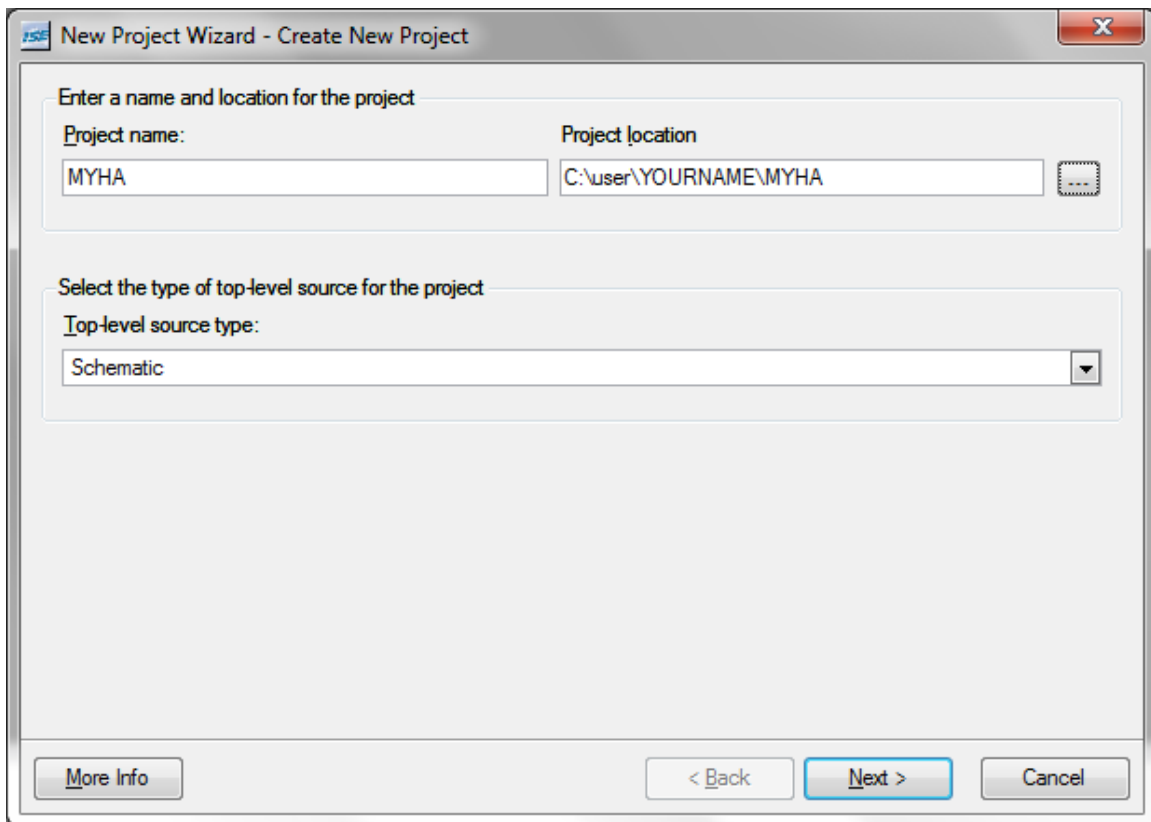
General Xilinx Tips

1. **SAVE EARLY AND OFTEN**
Xilinx is notorious for crashing at the most inopportune times. Do yourself a favor and save.
2. At the end of a lab session (or any work session), archive your project using the Xilinx utility (this will ensure you save everything), and save this zip archive on your ENIAC drive or on a flash drive. Do NOT assume files will remain on the lab computers or that “your” computer will be available at a later time.
3. Make sure all components are connected. Loose wires are a frequent cause of problems.
4. Try your hand at debugging first before calling a TA. You will learn a lot by struggling through problems that seem hard at first.
5. Read all instructions carefully before starting the lab. Often, there will be a little detail that ends up being very important.
6. Make sure you test all important cases, particularly edge/corner cases. You can be sure that your TA will test these as part of the demo.

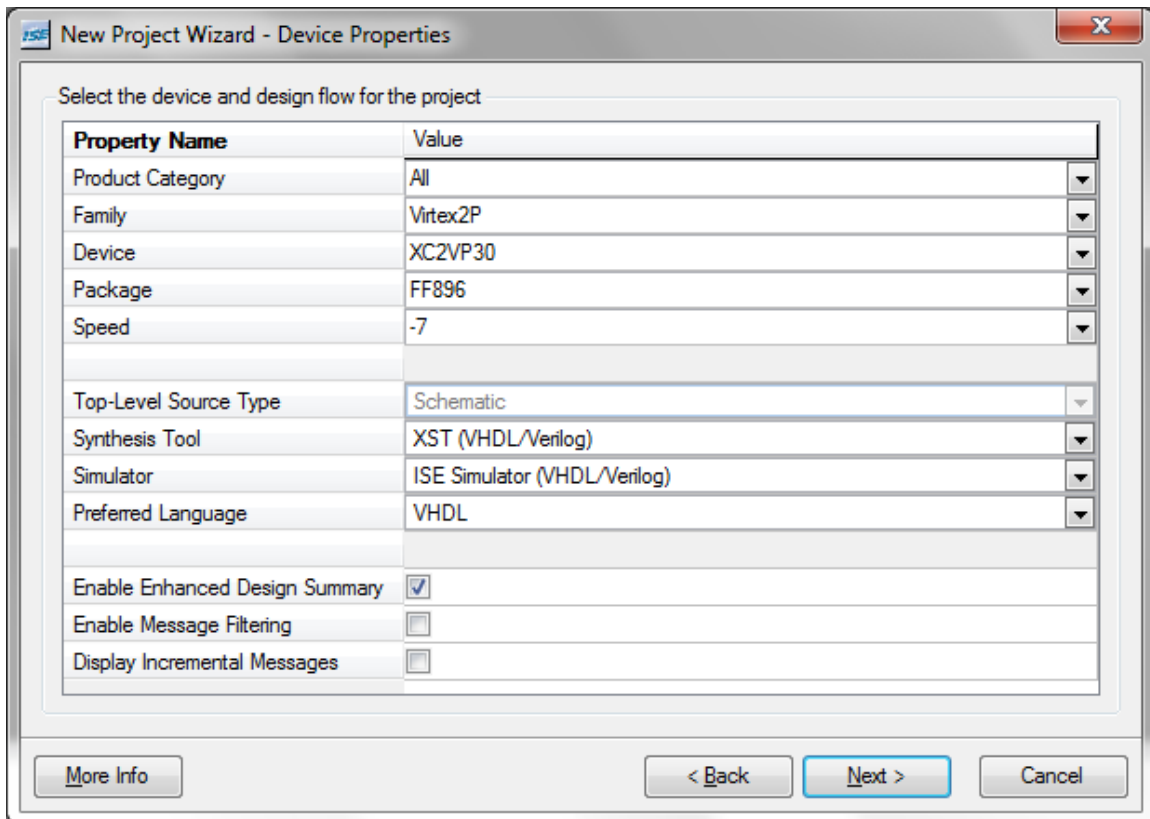
Creating a Project

First, you will have to create your project. Follow the steps below to start the program and create a new project.

1. Go to *Programs* → *ESE Lab Software* → *Xilinx ISE* → *ISE* and click on “Project Navigator”
2. Click on *File* → *New Project*
3. Before you name your project, type in the project location as “C:\user\yourname”
4. Name your project MYHA (this will append “MYHA” to your path)
5. Set the “Top-Level Source Type” to “Schematic”
6. Your screen should look like the image below.
7. Click “Next”



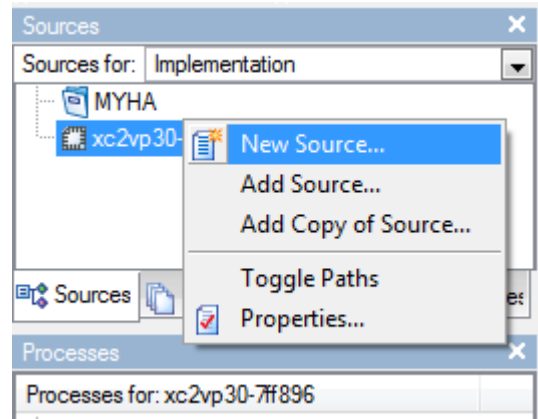
8. Set the device properties as shown in the screenshot below. You have to set them in order from top to bottom. Make sure you have ALL the settings correct before proceeding. You can change them later, but if they are wrong, certain things will not work.
9. Click “Next” a few times to go through the rest of the screens and finish the wizard. These screens allow you add sources to your project, but we will do this later.
10. Congratulations! You have created your first Xilinx project.



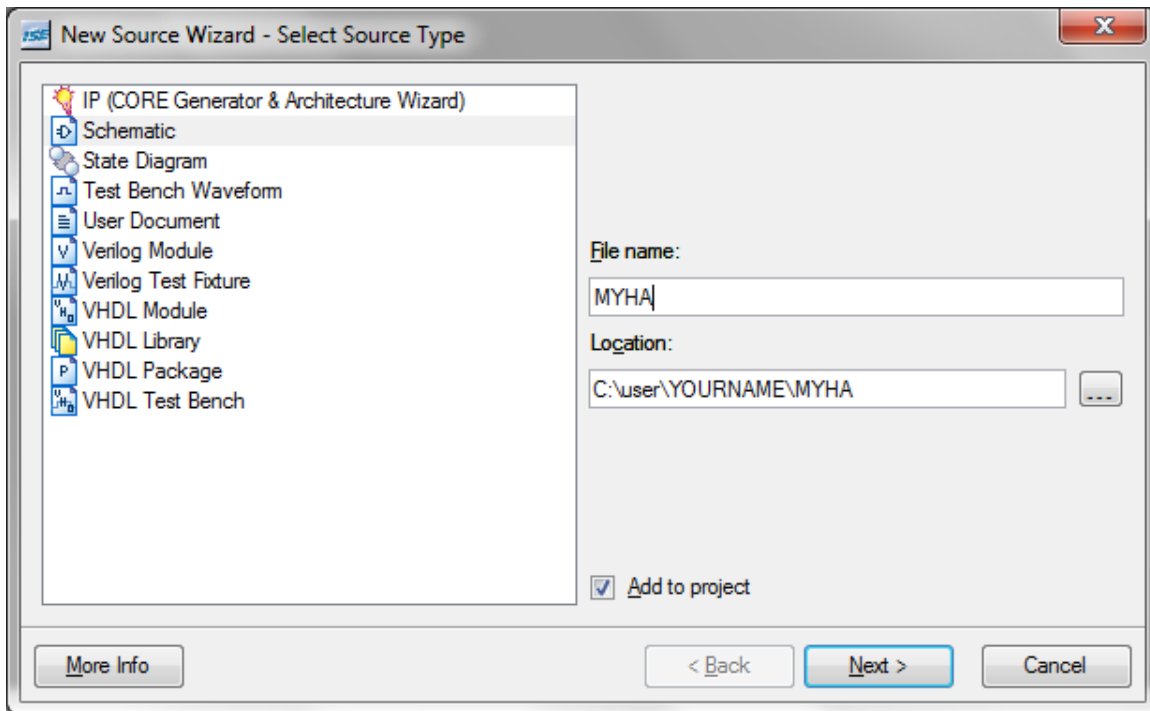
Schematic Entry

Now, we will create a schematic for the half-adder.

11. Right-click on “MYHA” in the Sources window on the top-left of the workspace, and click “New Source” (see screenshot below)

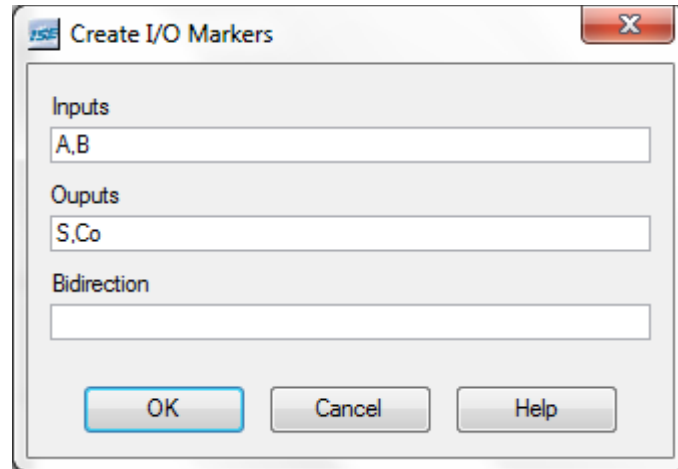


12. Select “Schematic” and type in “MYHA” for the filename (see screenshot below)



13. Click “Next” and then “Finish”
14. A blank schematic sheet should now appear.

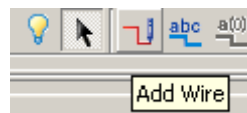
15. Now, we need to create I/O pins to interface with the half adder. Click on “Tools” in the menu bar, and select “Create I/O Markers”
16. We want two inputs (A and B) and two outputs (S for sum and Co for carry-out). Under “Inputs” type “A,B” and under “Outputs” (yes, it’s misspelled) type “S,Co”



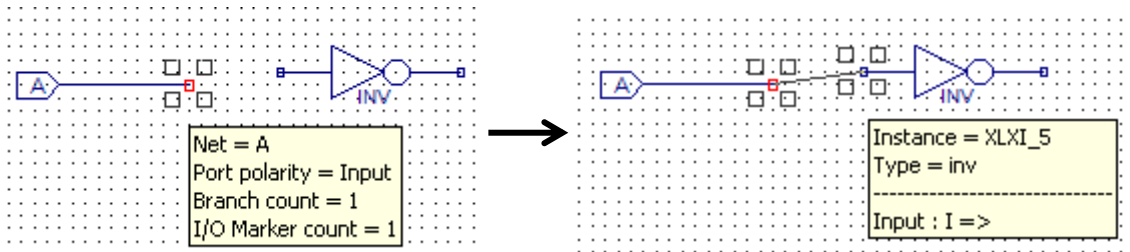
17. You can only use the wizard to add pins on an empty schematic. If you need to add pins after already adding other components to your schematic, use the “Add I/O Marker” tool in the toolbar, or press CTRL+G. You can only add an I/O marker to the end of a wire.



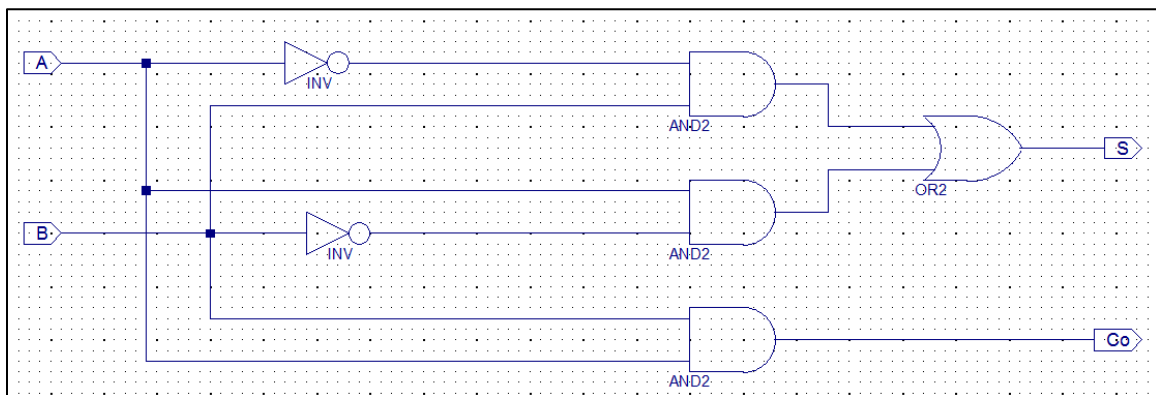
18. Next, we have to add the gates necessary for a half adder. The schematic from the first page is repeated below. The half adder uses two inverters (NOT), three AND gates, and one OR gate.
19. To add components, go to the “Symbols” menu in the “Sources” window
20. In the “Symbol Name Filter” blank, type in “inv”, click on the “inv” in the list, and click twice in the workspace (to add two inverters). Press the escape key to get out of symbol insert mode.
21. Now, add three “and2” modules and one “or2” module
22. Make sure you align your components so they’re roughly organized like the schematic. This will make it easier to make a clean schematic.
23. Now, it’s time to wire. Click on the wire tool in the toolbar or press CTRL+W (W for wire... shocking...)



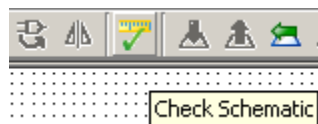
24. Left-click the spot you want to start a wire, and left-click on the lead of the component you want to connect it to. Xilinx will automatically lay the wire for you. If you want to manually make turns, left-click at intermediate spots, and it will place a junction there so you can “turn” the wire. Pressing the escape key will get you out of wire mode.



25. Make sure that your wire actually connected the two components. Occasionally, it will not fully connect, and usually leaves a red box indicating that a wire is not connected.
26. Use the wire tool to wire your schematic according to the schematic given above. Once done, you may want to move some components around to make the circuit clean and readable. This will be even more important as your circuits become more complex. Your schematic should roughly look like the one below.



27. Now that the schematic is complete, verify that the wires in your schematic are connected properly. Press the “Check Schematic” button on the toolbar.

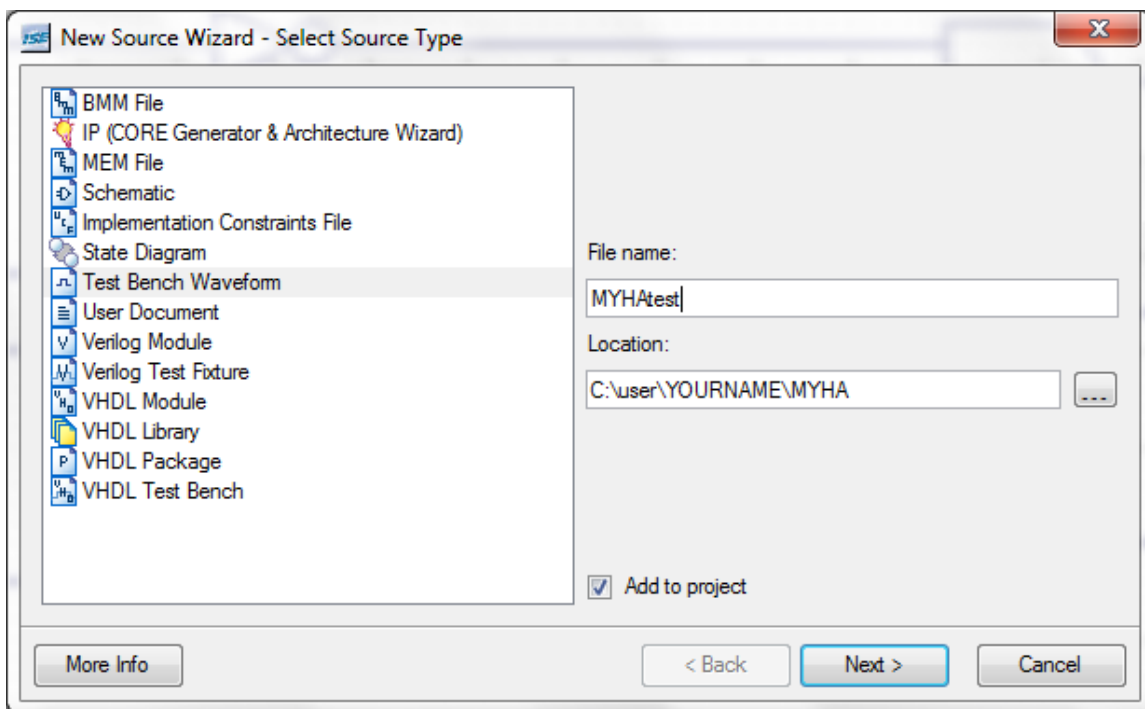


28. In the status menu below your schematic, it should say “No error or warning is detected”
29. Save (as you should anyway be doing frequently).

Behavioral Simulations (Testing)

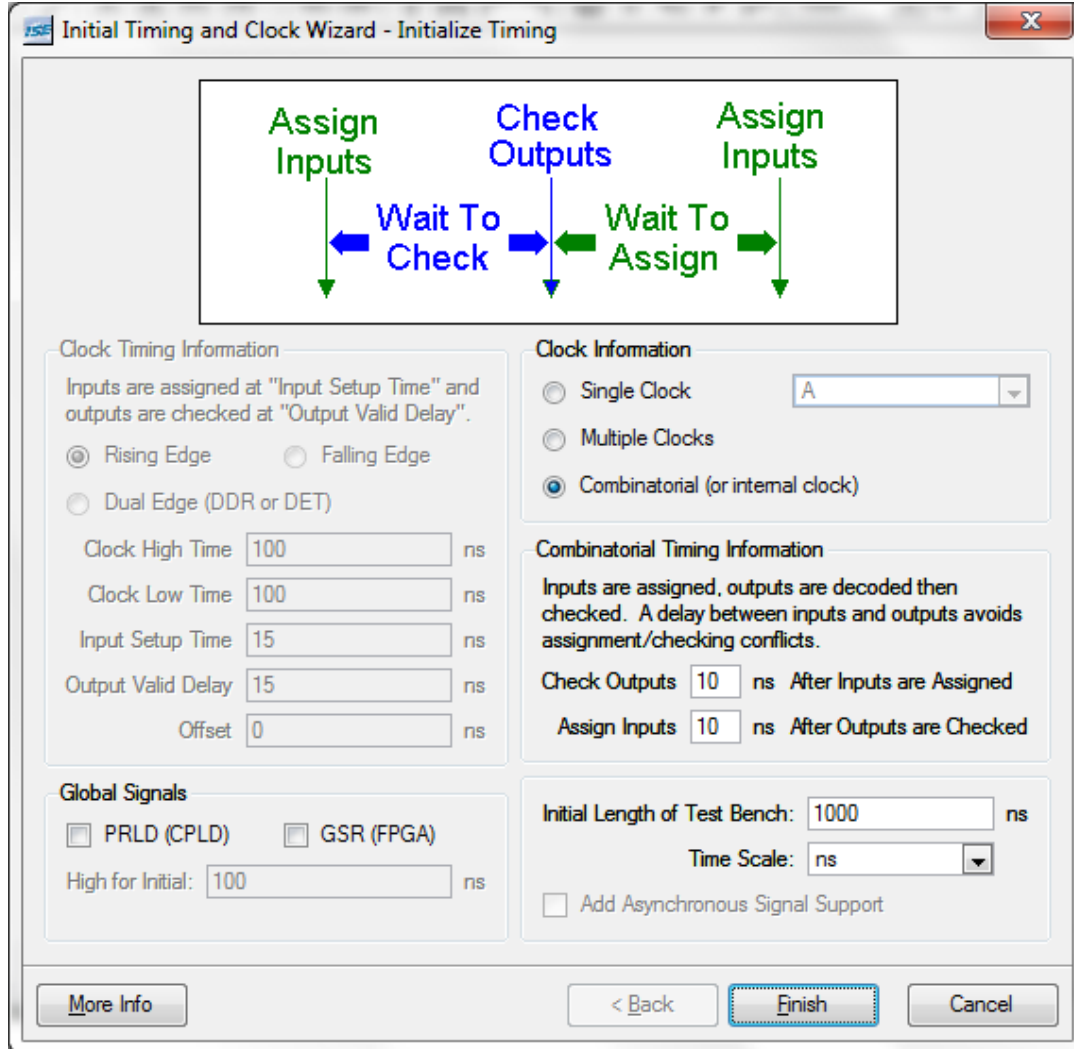
As with any software project, you have to test your design to see if it matches the theoretical expectations. For example, 0+0 should result in 0, not 42. To test a logic circuit in Xilinx, we run a behavioral simulation. This tests various input combinations to see if they generate the appropriate output.

30. In the Sources window, go to the drop-down menu labeled “Source for:” and select “Behavioral Simulation”
31. Right-click on the MYHA project in the Sources window and click on “New Source” as you did to add the schematic.
32. Select “Test Bench Waveform” and name it MYHAtest



33. Click Next. Associate your test bench with your MYHA schematic.
34. Click Next and then Finish.
35. After a few seconds, the “Initial Timing and Clock Wizard” window should appear
36. Select “Combinatorial” under clock, change the “Check Outputs” and “Assign Inputs” times to 10ns, and uncheck “GSR (FPGA)”

37. Your settings should match the screenshot below.

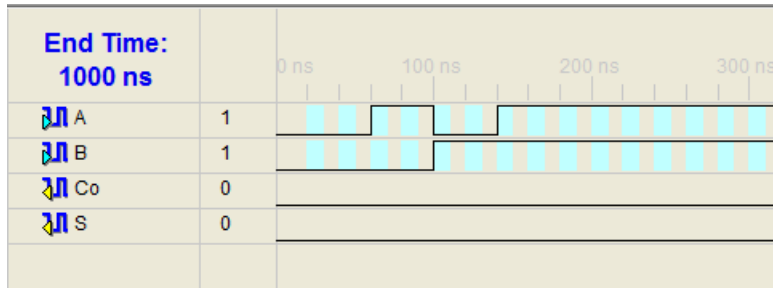


38. Click Finish

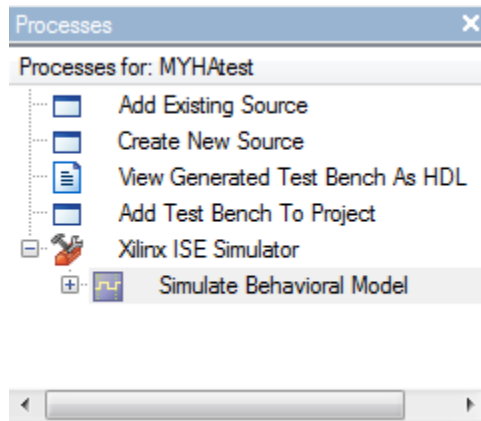
39. A test bench waveform should now appear. The blue shaded areas indicate the time after which the outputs will be valid (in our case we selected 10ns). By clicking on a blue space for the waveform of one of your inputs, you can change the state of that input between low (0) and high (1).

40. Now, to verify that our half adder works for all possible inputs, we need to test the adder for all possible inputs (AB=00, 01, 10, and 11). For more complex circuits, you probably will not be able to test all possible inputs. In those cases, you should test representative cases that, together, prove functionality of your circuit.

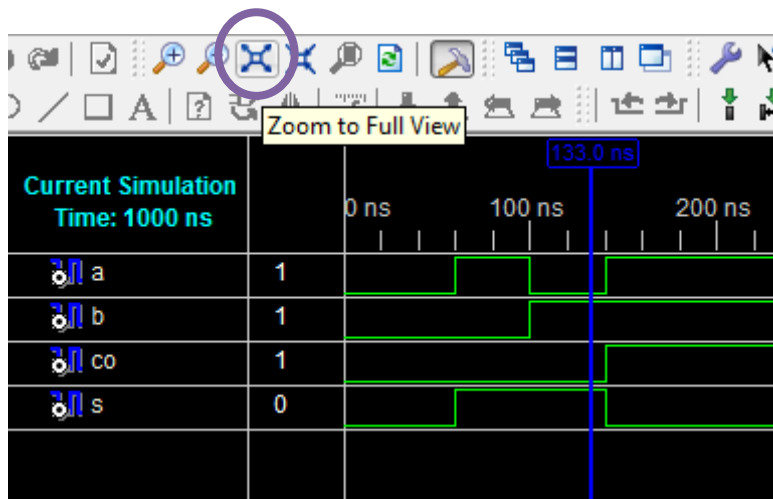
41. By clicking on the waveform, set it to input each of the possible inputs, as shown. You should leave each input steady for a decent period (say 50ns, or 5 cycles) to allow the output a chance to change values. Your waveform should look like the one below.



42. Save your test bench. The MYHAtest.tbw source should now appear in the Sources window. If you expand it, you will see the source it is linked to, which in this case is MYHA.sch.
43. It is now time to simulate! In the Processes window (below the Sources window), expand the “Xilinx ISE Simulator” menu and double-click “Simulate Behavioral Model”



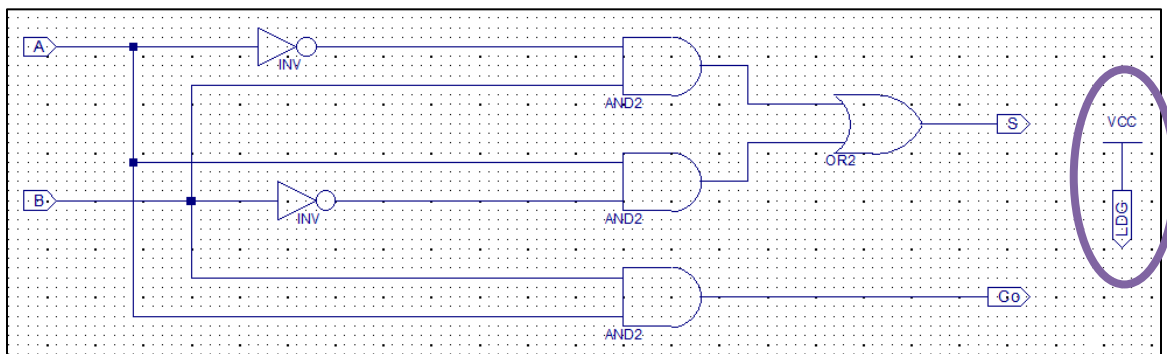
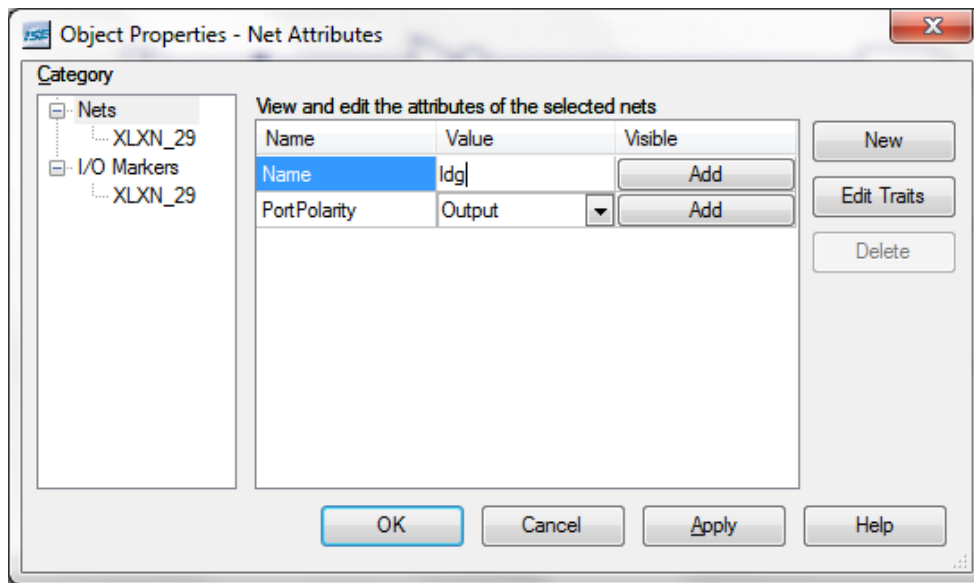
44. Click on the “Zoom to Full View” button on the toolbar (see screenshot). Your simulation should look like the image below.
45. As expected, the results correspond to the outputs we were expecting: AB=00 gives an output of CoS=00, 10 gives an output of 01, 01 gives an output of 01, and 11 gives an output of 10.



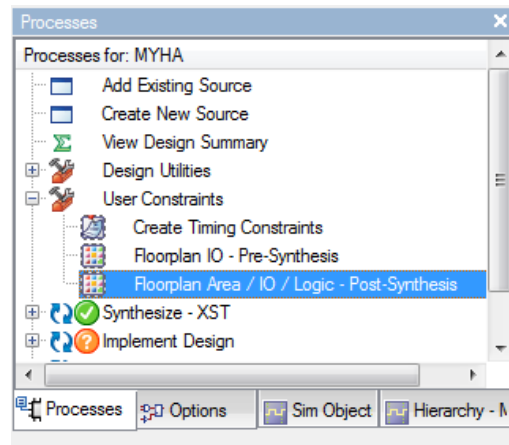
Implementation

Now we would like to implement the half adder on the FPGA board. This involves adding pin assignments to map the I/O pins in the design to pins on the FPGA board. Then, the Xilinx ISE program will synthesize the design for programming the board. This involves figuring out what LUTs (lookup tables) are needed, how the CLBs (configurable logic blocks) will be arranged, etc.

46. Switch back to “Implementation” mode by going to the “Sources for:” drop-down menu in the Sources window
47. In order to use the LEDs on the board, we have to add a pin called LDG. This corresponds to a flip-flop input on the FPGA chip that effectively turns on LED functionality.
48. Add the symbol “vcc” to the MYHA.sch schematic using the Symbols window. This can be placed anywhere.
49. Now, add an I/O marker to the end of the vcc component. Double-click on this I/O marker and name it “LDG” and make sure it is set to “Output” mode.



50. Next, we have to assign the pins to the I/O markers. Click on MYHA.sch in the Sources window. In the Processes window, click on “Floorplan Area / IO / Logic – Post-Synthesis”



51. Click “Yes” on the popup message to create a UCF (Implementation Constraint File) in the project
52. The Xilinx PACE program should now be open
53. A list of all the pins in your project should be listed on the left. In the “Loc” column, you have to enter the pin assignments for each one. Pin assignments are located on the ESE171 website ([link](#)). You may choose any switch/LED you want, but A and B should be switches, and S and Co should be LEDs. For each of the pins, a blue dot should appear on the Device Architecture window on the right side of the editor. Your assignments should look like the image below.

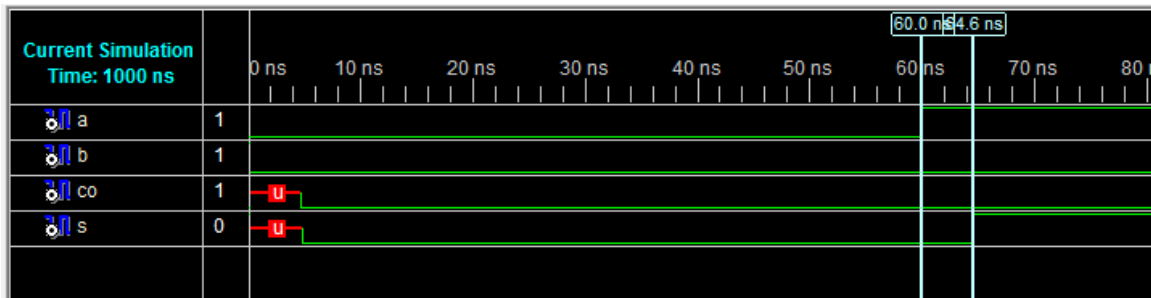
I/O Name	I/O Direction	Loc	Bank	I/O S
A	Input	N6	BANK	
B	Input	L5	BANK	
Co	Output	T8	BANK	
LDG	Output	V7	BANK	
S	Output	U5	BANK	

54. Save the file and close the window. If a window asking you to select a Bus Delimiter appears, select XST default and click OK.
55. Now, you can implement the design. To run through the full synthesis and implementation sequence, double-click on “Generate Programming File” in the Processes window. This will generate the Translate, Map and Place & Route files. A green checkmark will appear next to each step when the process has finished successfully. If any red X appears, then there was an error. At this stage, ask a TA for help.

Timing Simulation

At this point, you can verify your design by doing a Post-Place & Route Simulation. Since the design has been placed on the chip, the system can do a more accurate simulation that will also give you specific timing information such as the delays of the output signals.

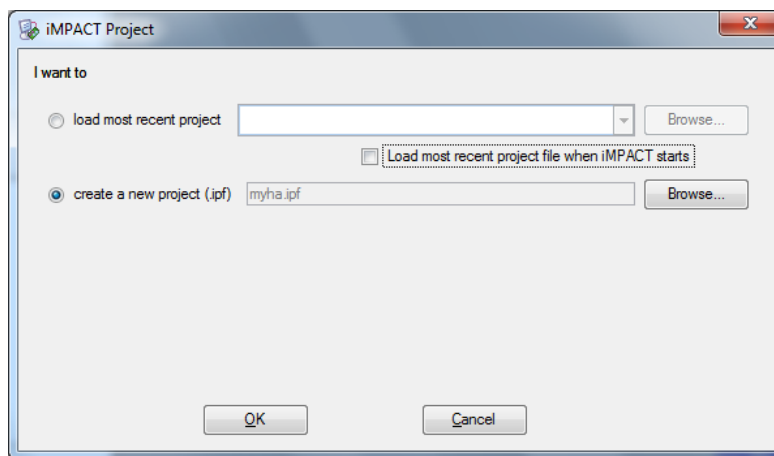
56. In the Sources window, go to the “Sources for:” drop-down menu and select “Post-Route Simulation”.
57. Select the MYHAtest.tbw source
58. In the Processes window, expand the Xilinx ISE Simulator menu and double-click “Simulate Post-Place & Route Model”
59. This will produce a simulation very similar to the behavioral simulation you did before. Now, however, the outputs will appear with some delay after the inputs change. You can measure this delay by zooming in on a specific region and adding markers. This is very similar to measuring analog circuit delay on an oscilloscope.
60. Right-click on the simulation window and select “Add Marker”
61. Click on a rising/falling edge of the input you want to measure delay from
62. Repeat step 62 and add a marker to the rising/falling edge of the output that corresponds to your marked input
63. You can now calculate the delay by simply subtracting the two times
64. A sample measurement is shown below



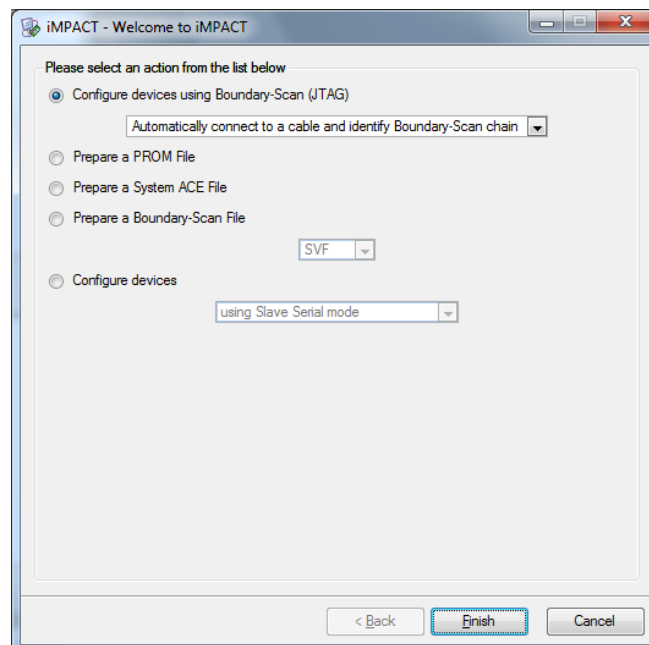
Programming the Board

It is finally time to download the circuit to the board!

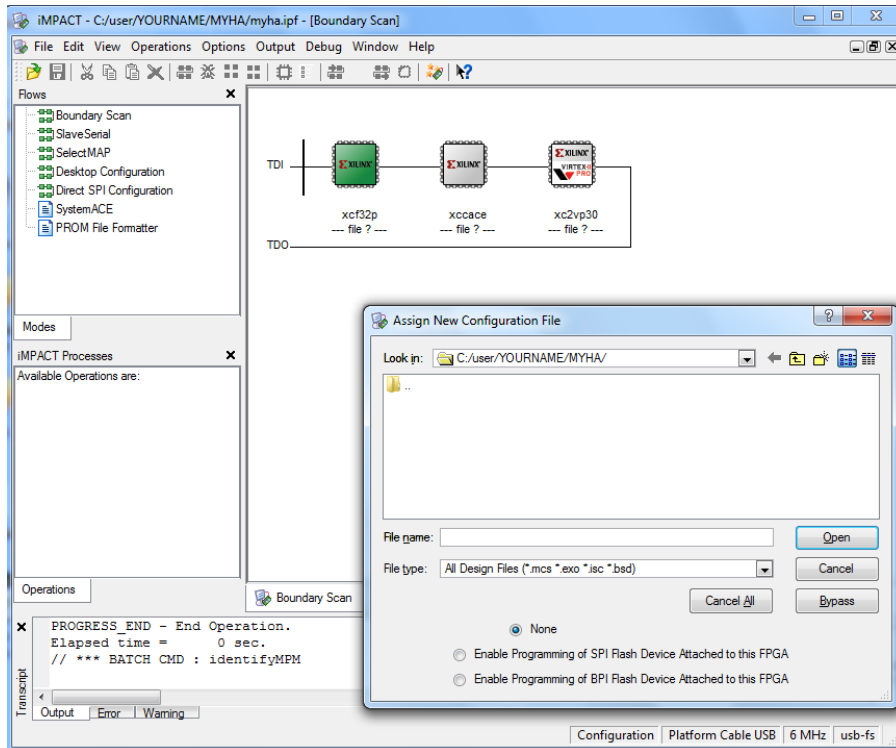
65. Power the board and connected the USB cable to the computer
66. Turn on the board
67. Open the Windows Start Menu, and in the search field, type “impact”
68. Click on the program “iMPACT (64-bit)”
69. This will open the iMPACT program, which you will use to download the programming file (.bit file) to the board.
70. Select “create a new project (.ipf)” and browse to the directory where you saved your project (C:\user\YOURNAME\MYHA), and name the file myha.ipf



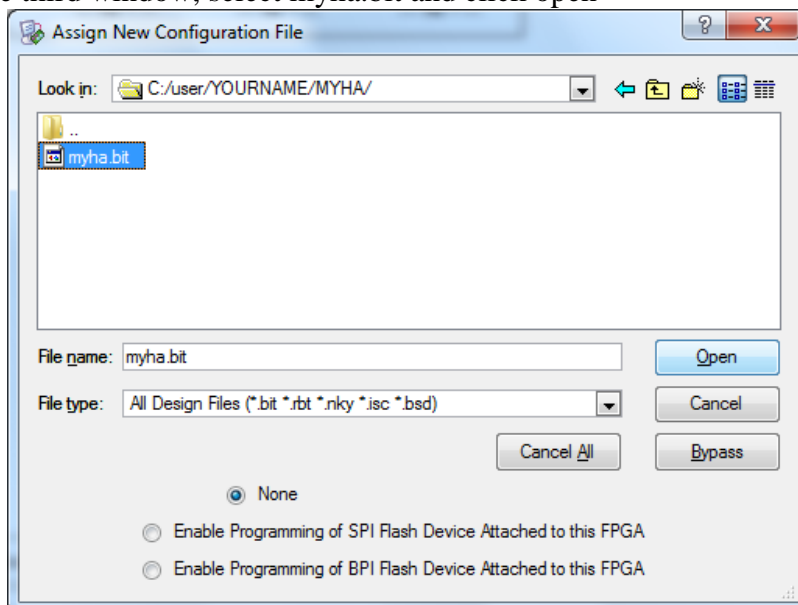
71. Click ok to create the iMPACT project
72. Select “Configure devices using Boundary-Scan (JTAG)” on the next screen



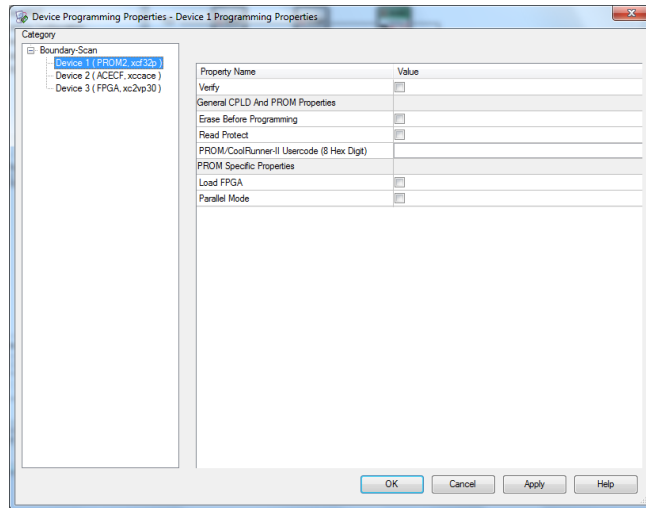
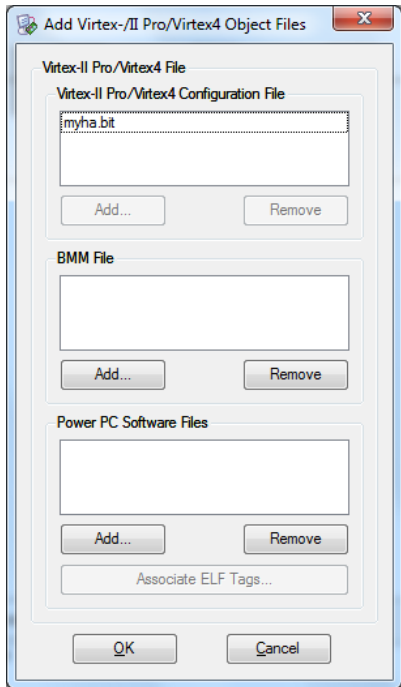
73. It will now find and load the board configuration
74. You should now see a chain of three devices you can program and a pop-up window. Each device programs a different part of the board, like the chip itself and the PROM (programmable read-only memory).



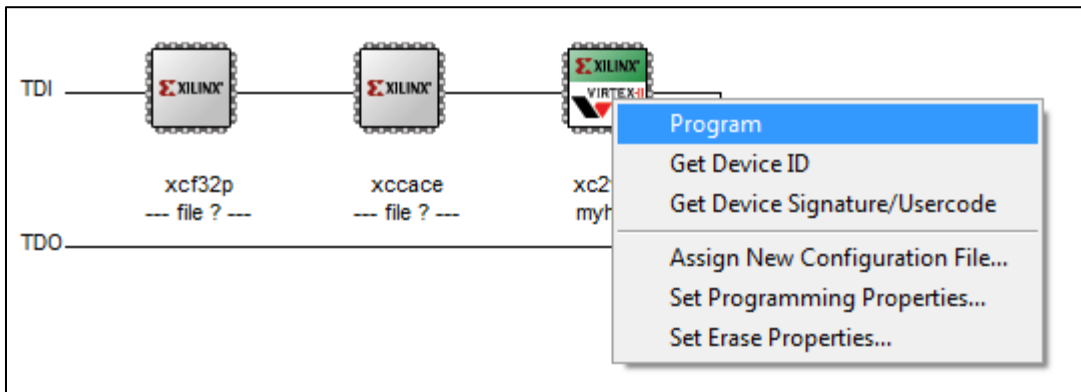
75. Click cancel on the first pop-up window, and then again on the second pop-up window, to skip the first two program locations
76. On the third window, select myha.bit and click open



77. Click OK on the next two windows



78. Right-click on the xc2vp30 chip in the window (the third one) and select Program



79. You should see a blue pop-up message saying “Program Succeeded”

80. Test the functionality of the half adder on the board

81. Congratulations! You have completed your first Xilinx project!

Snapshots and Archives

At any time, you can take a snapshot of your project. This is a *read-only* copy of the current project, similar to taking an image of your computer as a back-up. This allows you to save different versions of your project in the event something corrupts your project in the future. A snapshot includes all files and directories associated with a project.

1. Go to Project → Take Snapshot
2. Name the snapshot (keep it within eight characters) and add a comment if you'd like that describes it.
3. Snapshots can be viewed by going to the Snapshots tab on the Sources window.
4. You can revert to a snapshot by right-clicking on it and clicking “Make Snapshot Current”

You can also make an archive of a project. This creates a zip file that contains all files and directories associated with your project. You cannot open an archived project from within ISE Project Navigator. To restore an archived project, you first need to unzip it (using a Windows unzipping utility). Then, open the project as usual by clicking on the .ise file within the unzipped project folder, or open the .ise file from Project Navigator by going to File → Open Project.