

# Lab 1: Full Adder

## Introduction

In this lab you will design a simple digital circuit called a *full adder*. You will then use logic gates to draw a schematic for the circuit. Finally, you will verify the correctness of your design by simulating the operation of your full adder. This lab should also provide you with an understanding of how to use the *Xilinx ISE 4 (Project Navigator)* software.

This lab is divided into three parts: design, schematic, and simulation. The first part can be done on paper, but remainder of the lab will require the use of a PC with access to the Xilinx tools.

## Background: Adders

An adder, not surprisingly, is a circuit whose output is the binary sum of its inputs. Since adders are needed to perform arithmetic, they are an essential part of any computer. In future labs, you will be using the adder that you design in this lab to perform arithmetic in a working microprocessor.

A full adder has three inputs (A, B, CARRY\_IN) and two outputs (SUM, CARRY\_OUT), as shown in Figure 1. Inputs A and B represent bits of the two binary numbers that are being added, and SUM represents a bit of the resulting sum.

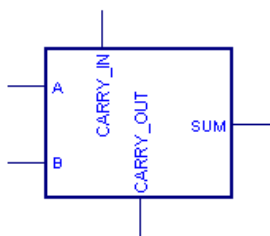


FIGURE 1 Full Adder.

The CARRY\_IN and CARRY\_OUT signals are used when adding numbers that are more than one bit long. To understand how these signals are used, consider how you would add the binary numbers 101 and 001 by hand:

$$\begin{array}{r} 1 \\ 101 \\ + \quad 001 \\ \hline 110 \end{array}$$

You first add the two least significant bits. Since  $1+1=10$  (in binary), you place a zero in the least significant bit of the sum and carry the 1. Then you add the next two bits with the carry, and place a 1 in the second bit of the sum. Finally, you add the most significant bits (with no carry) and get a 1 in the most significant bit of the sum.

When a sum is performed using full adders, each adder handles a single column of the sum. Figure 2 shows how to build a circuit that adds two three-digit binary numbers using three full adders. The CARRY\_OUT for each bit is connected to the CARRY\_IN of the next most significant bit. Each bit of the three-bit numbers being added is connected to the appropriate adder's inputs and the three SUM outputs make up to full 3-bit sum result.

Note that the leftmost CARRY\_IN input is unnecessary, since there can never be a carry into the first column of the sum. This allows us to use a half adder for the first bit of the sum. A half adder (see Figure 3) is similar to a full adder, except that it lacks a CARRY\_IN and is thus simpler to implement. To save your design time, however, we will only use full adders in this lab.

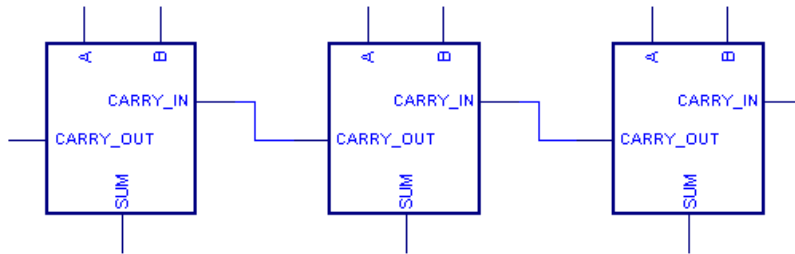


FIGURE 2 3-Bit Adder.

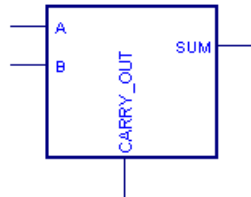


FIGURE 3 Half Adder.

## Design

A partially completed truth table for a full adder is given in Figure 4. The table indicates the values of the outputs for every possible input, and thus completely specifies the operation of a full adder. As is common, the inputs are shown in binary numeric order. The values for SUM are given, but the CARRY\_OUT column is left blank.

- Complete the table by filling in the correct values for CARRY\_OUT so that adders connected as in Figure 2 will perform valid addition.

The SUM output can be produced from the inputs by connecting two XOR gates as shown in Figure 5. You should convince yourself that this circuit produces the outputs for SUM as given in the table.

- Using only two-input logic gates (AND, OR, XOR) and inverters (NOT), design a circuit that takes A, B, and CARRY\_IN as its inputs and produces the CARRY\_OUT output, as shown in Figure 6. Try to use the fewest number of gates possible. For now, you may have to use trial and error to find a set of gates that works. We will soon be learning a systematic way to do this using Boolean algebra.

Inputs			Outputs	
CARRY_IN	B	A	CARRY_OUT	SUM
0	0	0		0
0	0	1		1
0	1	0		1
0	1	1		0
1	0	0		1
1	0	1		0
1	1	0		0
1	1	1		1

FIGURE 4 Partially completed truth table for a full adder.

## Schematic

Now that you know how to produce both the SUM and CARRY\_OUT outputs using simple logic gates, you might want to construct a working full adder circuit using real hardware. One way to do this is to enter the schematic representation of your logic into a software package that is capable of programming the schematic into an

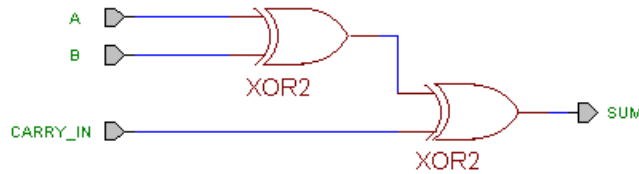


FIGURE 5 Schematic for SUM logic.



FIGURE 6 Schematic of CARRY\_OUT logic (for you to complete).

integrated circuit. We will be using the Xilinx ISE 4 software for these purposes. The Xilinx software is a powerful and popular commercial suite of applications used by hardware designers. You will need a Windows computer with this software installed.

First, you will learn how to start a new project in Xilinx. Enter the Xilinx software by clicking on the icon for “Project Navigator,” found on the desktop of the Engineering Server. Or you can select Start→Program→Xilinx ISE 4→Project Navigator. This will open the *Project Navigator*, shown in Figure 7, which will always serve as your entry point into the Xilinx applications.

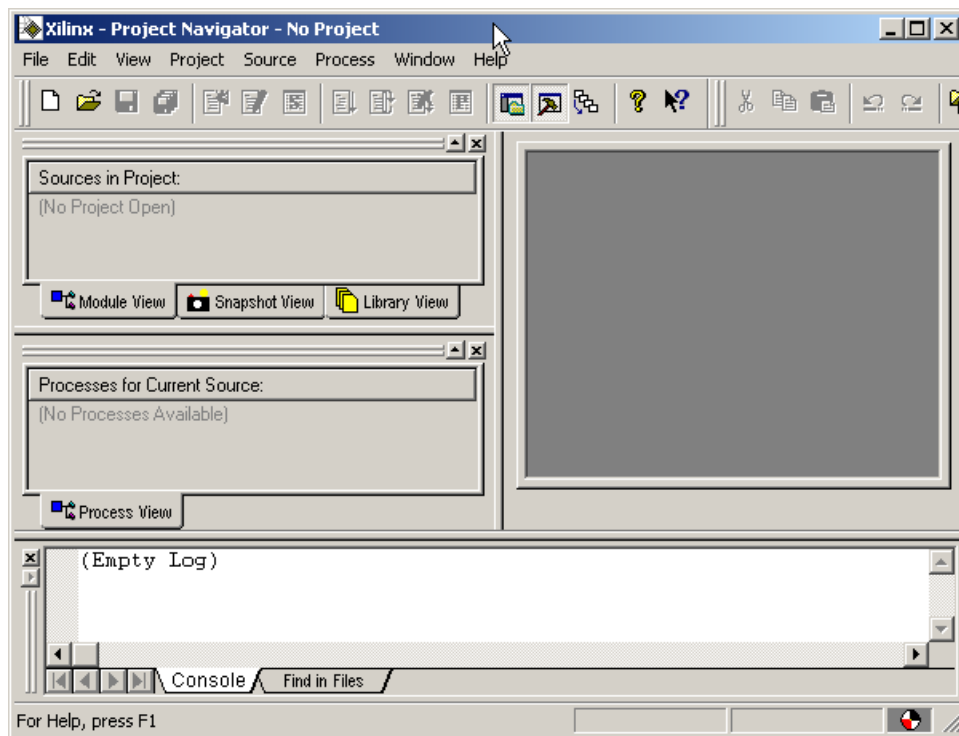


FIGURE 7 Xilinx project navigator.

In the Project Navigator window, choose File→New Project from the pull-down menu. In the New Project dialog window shown in Figure 8, type the desired location in the Project Location field, or browse to the directory under which you want to create your new project directory using the browse button next to the Project Location field. The next step is to enter “lab1\_xx” (where xx are your initials) as the name of your project. Use the pull-down arrow to select the value for each property of Project Device Options. Select the device family to be Virtex

or Virtex II, then select XST Verilog as the Design Flow and leave the Device as the default value. When you have finished all this, press OK to begin your first Xilinx project now.

Now you are ready to create a schematic for your project. In the Project Navigator window, choose Project→New Source from the menu. A New window will pop up, as shown in Figure 9.

Select Schematic in the list, and type “LAB1\_XX” as the name of your schematic file. This file again will be stored in the directory you specified. Make sure you check the Add to project box. When you are done, click Next and then click Finish. An ECS schematic editor window with a blank sheet will show up, as in Figure 10. The schematic editor has an interface similar to many drawing programs. You can drag and drop logic gates and draw wires between them. This is where you will draw and connect the logic gates to build your full adder circuit.

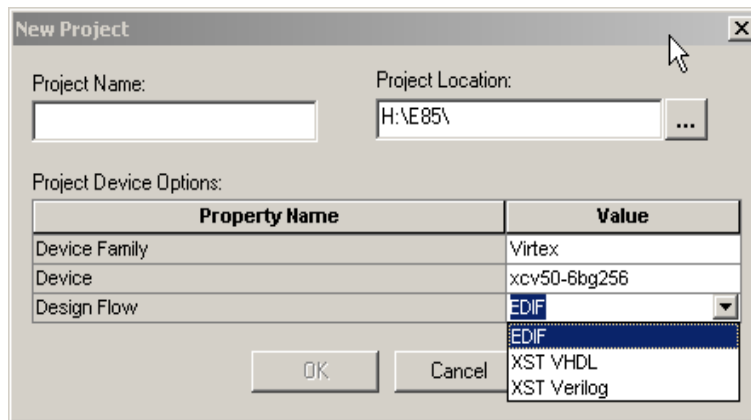


FIGURE 8 New project window.

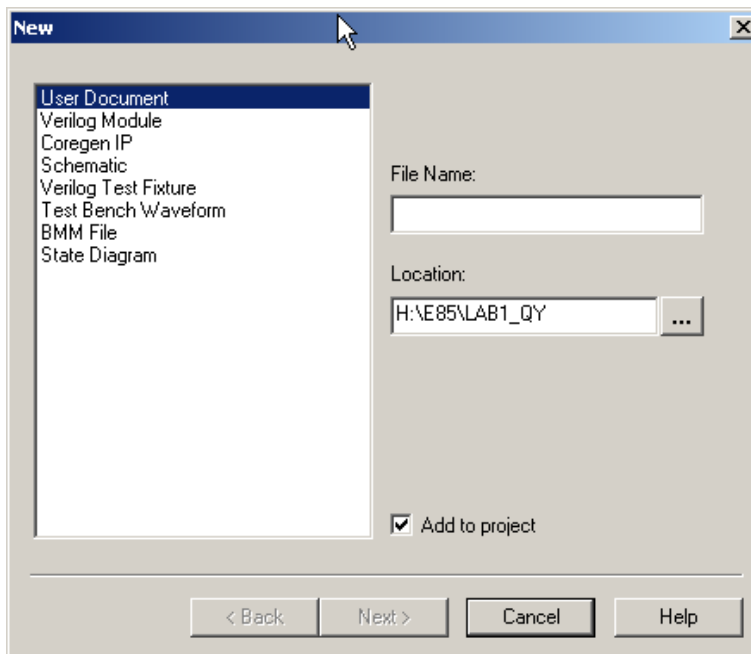


FIGURE 9 New source window.

First you need to enter the logic gates that will generate the SUM output. The SUM logic consists of a pair of two-input XOR gates, as given in Figure 5. To draw these gates, first select Add→Symbol in the menu or click on the Add Symbols button in the tool bar. This opens a window (see Figure 11) that lists the available circuit symbols.

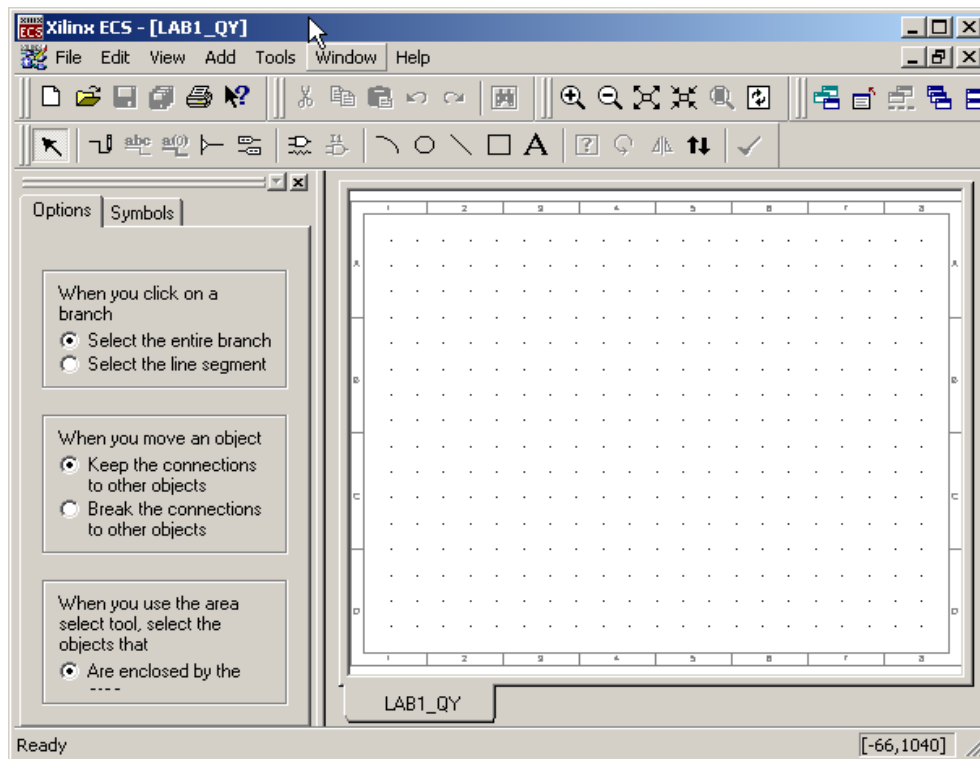


FIGURE 10 Xilinx Schematic Editor.

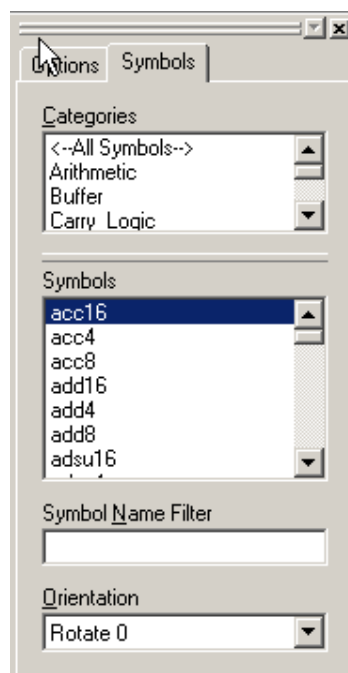


FIGURE 11 Circuit symbols window.

You'll notice that the library of symbols is quite large. The available symbols range from the simple logic gates that we will be using for now to more complex circuit components. Scroll down and select the xor2 symbol. (Al-

ternatively, you can type “xor2” in the Symbol Name Filter at the bottom of the window.) You can select the orientation of the symbol using the pull-down button.

If one item (e.g. xor2) in the symbol list is highlighted, your mouse cursor will now appear differently when you move to the sheet. You can also press Ctrl+R on the keyboard to rotate the symbol. Simply click at the location where you want to place the xor2 gate on the sheet. Place another xor2 gate in the proper location. If you are unsatisfied with the placement of objects in the schematic, at any time you can drag objects to new locations by choosing the Select tool.

You are now ready to wire up this part of the circuit. Select Add→Wire from the menu or click on the Add Wire button. Wires can be drawn in much the same way as you would draw lines in a standard drawing program. You can click on one end to begin drawing a wire, and drag the wire until you release the click at the other end of the wire.

Once you finish all the connections between the symbols, you are ready to define the terminals. Choose Add→I/O Marker from the menu or click on the Add I/O Marker button. Select Add an input marker and select Left as the orientation of the marker. Move the cursor close to the end of the wire for input A, and click when a text box is shown. In the select mode, by double clicking the marker, you can change the name of the terminal to A. *Note that the name of the port must start with a letter.* Properties of other objects in the schematic can also be edited this way. Now do the same thing to draw the B and CARRY\_IN terminals. Similarly select Add an output marker for the SUM terminal.

The software attempts to route wires intelligently for you. If you use the Select tool to move components that have wires connected to them, the wires will stay connected. You can also use the Select tool to drag sections of a wire that need to be re-routed, or to disconnect the end of a wire from a component and reconnect it elsewhere. Go ahead and add wires now so that your sheet looks like the one in Figure 5.

You are now ready to complete your schematic of the full adder by drawing the logic for CARRY\_OUT that you designed earlier. Add an output terminal for CARRY\_OUT and draw the necessary logic gates and wires to complete the circuit. The symbols you should use to draw your logic gates are as follows:

Type of Logic Gate	Symbol Name
And	And2
Or	Or2
Exclusive Or	Xor2
Not	Inv

Do not add a second set of input terminals for A, B and CARRY\_IN. Instead, note that you can connect multiple wires to the same input terminals (or you can connect wires to other wires to create branches).

This completes your full adder schematic and your introduction to the Schematic Editor. Each time you finish the schematic, remember to save the file. It is also necessary to check for any errors in the schematic before you do the simulation. Select Tools→Check Schematic in the schematic editor and make sure you correct all the errors. You may want to print out your schematic now, since you are required to turn it in. To do so, go to File→Print. Finally, choose File→Save to save your schematic before you exit the editor.

Now your Project Navigator window will show up again with a list of files of the project. If you want to revisit the schematic files, you can always double click the file in the source directory in the upper left pane, as shown in Figure 12.

## Simulation

One motivation for having entered your full adder circuit into the Xilinx software is that you can now use the software to simulate the operation of the circuit, for example, to verify the correctness of your design without actually building the circuit in hardware. In this part of the lab, you will first create a testbench waveform for your design using HDL Benchner, and then simulate the design using ModelSim.

The first step is to create a testbench waveform source. From the Project Navigator window, select Project→New source. In the New dialog window, select Test Bench Waveform as the source type and type the name “LAB1” or “TESTLAB1”. Now select Next and then select Finish. In the Initialize Timing window, use the

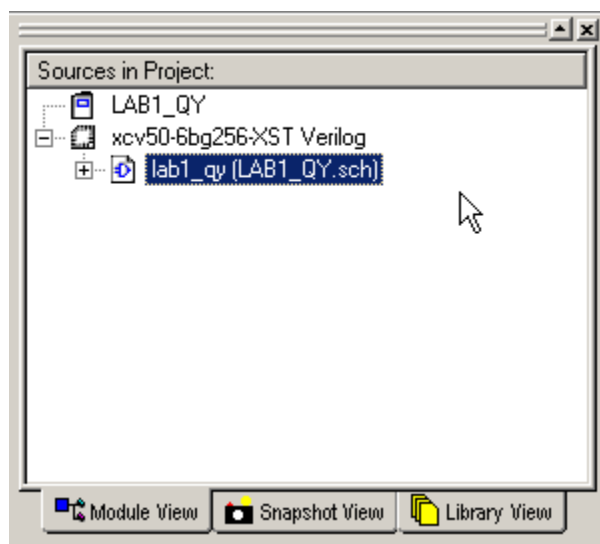


FIGURE 12 Project navigator window after schematic input.

default timing constraints for the testbench waveform. Now, HDL Bench will be launched, as shown in Figure 13.

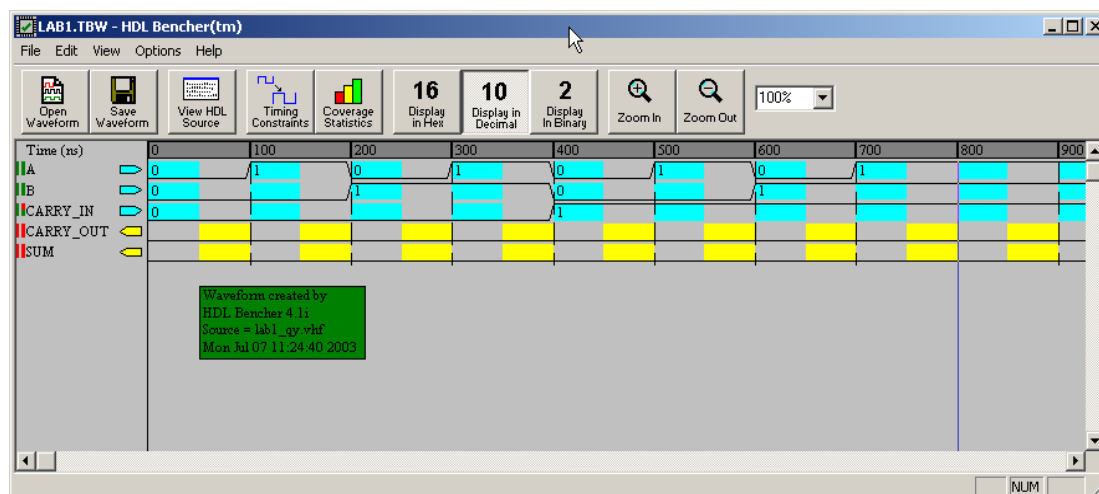


FIGURE 13 HDL Bench window.

Note that, before initializing the inputs, you should ensure that you have chosen View→Decimal or selected the 10 Display in Decimal button in the HDL Bench toolbar. You can then enter the input stimulus in the blue area in each cell. For example, click cell A so that it is set high at even cycles 2, 4, 6 and 8. You can drag the blue vertical line to set cycle 8 to be the end of the test. Similarly for input B and CARRY\_IN: B is set high at cycles 3, 4, 7 and 8, while CARRY\_IN is set high at cycles 5, 6, 7 and 8. Now you can save the testbench waveform by selecting File→Save waveform or by clicking the Save icon before exiting HDL Bench.

In the Project Navigator window, you will see that the *lab1.tbw* file is listed as one of the source files associated with the schematic file. Select the *lab1.tbw* file, and you will see the ModelSim Simulator in the Processes for Current Source window. You can expand the option by click the left side “+”. In the list, as shown in Figure 14, double click Simulate Behavioral Verilog Model.

When ModelSim is launched, you are able to see the waveform in the time domain. If there is an error in launching ModelSim, look for error messages and correct them. Click the Zoom Full icon in the taskbar to see the whole waveform of the simulation results. Both input and output bus values can be set to be displayed in either

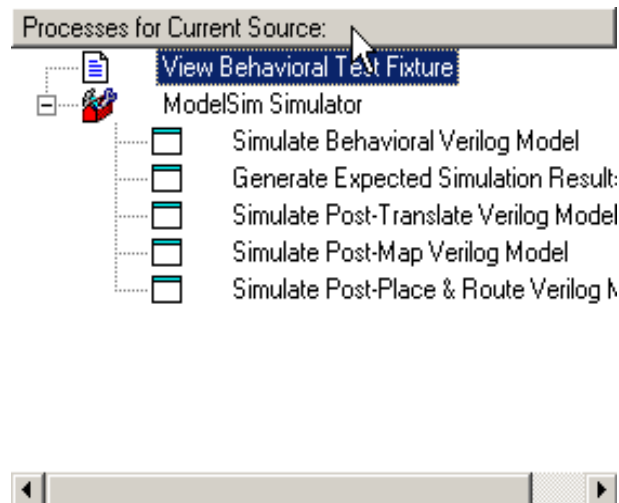


FIGURE 14 Choose Simulate Behavioral Verilog Model from ModelSim Simulator.

Hex, Decimal or Binary. Choose Radix→Binary/Decimal/Hexadecimal. When the output values are correct, you have a working full adder! Choose File→Print to print a copy of your waveforms to turn in. If the result is not what you expected, go back to the schematic and testbench waveform files and check the design. Congratulations on your completion of lab 1!