Introduction to Computer Engineering (E85)

Lab 1: Full Adder

Lab Overview

The only way to truly learn computer engineering is to do it. A key part of your Computer Engineering experience will be the series of labs. The following table summarizes the contents and learning objectives of the labs to give you the big picture. You may want to refer back to it from time to time through the course to see where you have been and where you are going.

Lab	Project	Learning Objectives
1	Full Adder	Schematic editor, simulator, combinational circuits
2	MIPS Controller	combinational circuit design
3	Adventure Game	hierarchy, Finite State Machine design
4	Thunderbird Tail Lights	FSMs, hardware implementation
5	32-bit ALU	hierarchy, arithmetic unit design
6	Fibonacci Numbers	PCSpim, assembly language programming
7	Floating Point Addition	assembly language programming, floating point
8-9	MIPS Processor	microarchitecture
10-11	Multicycle Processor	microarchitecture, independent design

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** software that we will be using throughout the semester.

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 on the Engineering Server. After completing the lab, you are required to turn in something from each part (refer to the "What to Turn In" section at the end of this handout).

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.

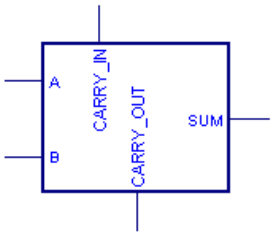


Figure 1: Full Adder

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:

```
  1
 101
+ 001
 110
```

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 in connected to the appropriate adder's inputs and the three SUM outputs make up to full 3 bit sum result.

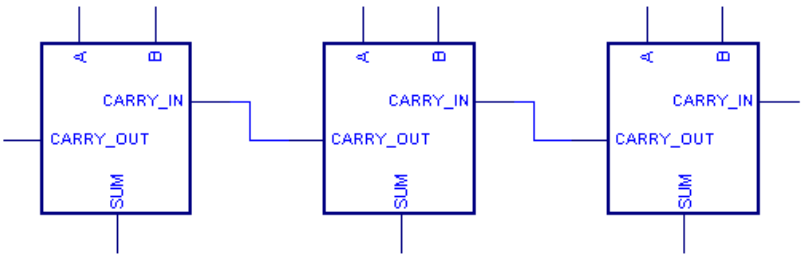


Figure 2: 3-bit Adder

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 is the 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.

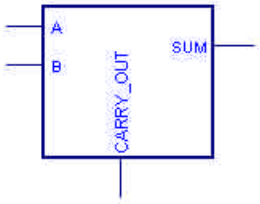


Figure 3: Half Adder

1. Design

A partially completed truth table for a full adder is given in Table 1. 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.**

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

Table 1: Partially Completed Truth Table for Full Adder

The SUM output can be produced from the inputs by connecting two XOR gates as shown in Figure 4. You should convince yourself that this circuit produces the outputs for SUM as given in the table.

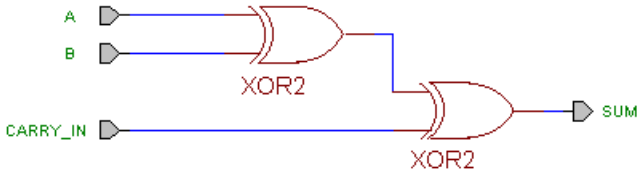


Figure 4: Schematic for SUM logic

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. 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.

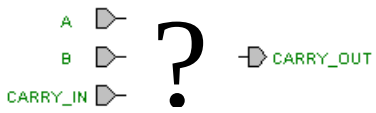


Figure 5: Schematic of CARRY_OUT logic (for you to complete)

2. 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 integrated circuit. This semester we will be using the Xilinx software for these purposes. The Xilinx software is a powerful and popular commercial suite of applications used by hardware designers. To use run the software you will need a Windows computer with access to the Engineering Server.

You will now learn how to use the Xilinx **Schematic Editor** to draw a schematic of your full adder. Enter the Xilinx software by clicking on the icon for "Xilinx Foundation Project Manager," found on the Engineering Server. This will open the **Project Manager**, shown in Figure 7, will always serve as your entry point into the Xilinx applications.

As soon as the Project Manager opens, a "Getting Started" window should appear. Select "Create a New Project" in this window and press OK. (If the "Getting Started" window did not appear, choose File•New Project in the Project Manager.)

You should now see a "New Project" window. Enter "LAB1_XX" (where XX are your initials) as the name of your project. Enter an appropriate location for your project in the Directory field (this should be an empty directory on your EngServe students directory). Choose "Foundation Series v2.1i" as the project Type and select "Schematic" as the Flow. Then in the pulldown bars at the bottom of the window, select "XC4000XL," "4010XLPC84," and "09" from left to right (these values are not relevant yet, so don't worry about what they mean). When you have finished all this, press OK to begin your first Xilinx project now.

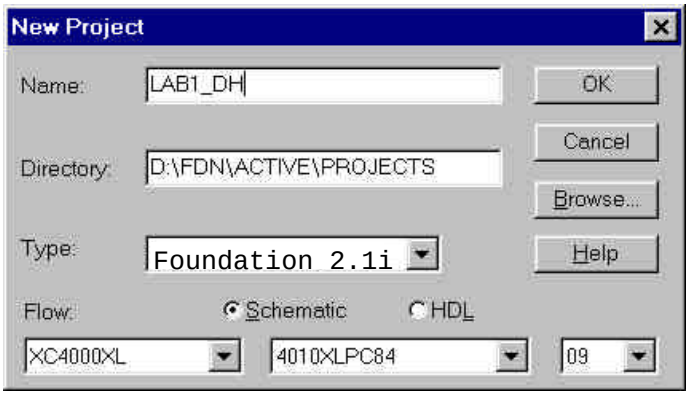


Figure 6: New Project Window

Take a moment to familiarize yourself with the Project Manager window, which is divided into three main panes:

- The upper left pane lists the files associated with the current project (which have been placed in the directory that you specified). There are several different types of files that can be part of a project. Each has a different filename extension, and displays with a different icon. Files can be opened by double-clicking on their name.

- The upper right pane contains a flow diagram that graphically depicts the status of the current project. Each box in the diagram corresponds to a different project phase. The arrows indicate the logical order in which projects can progress. The first stage of any project is the initial design entry, which is what you will be doing in this part of the lab.
- The lower pane contains a "Console," in which all status and error messages are displayed. Normal status messages are displayed in black text, while warning and error messages will be highlighted in different colors.

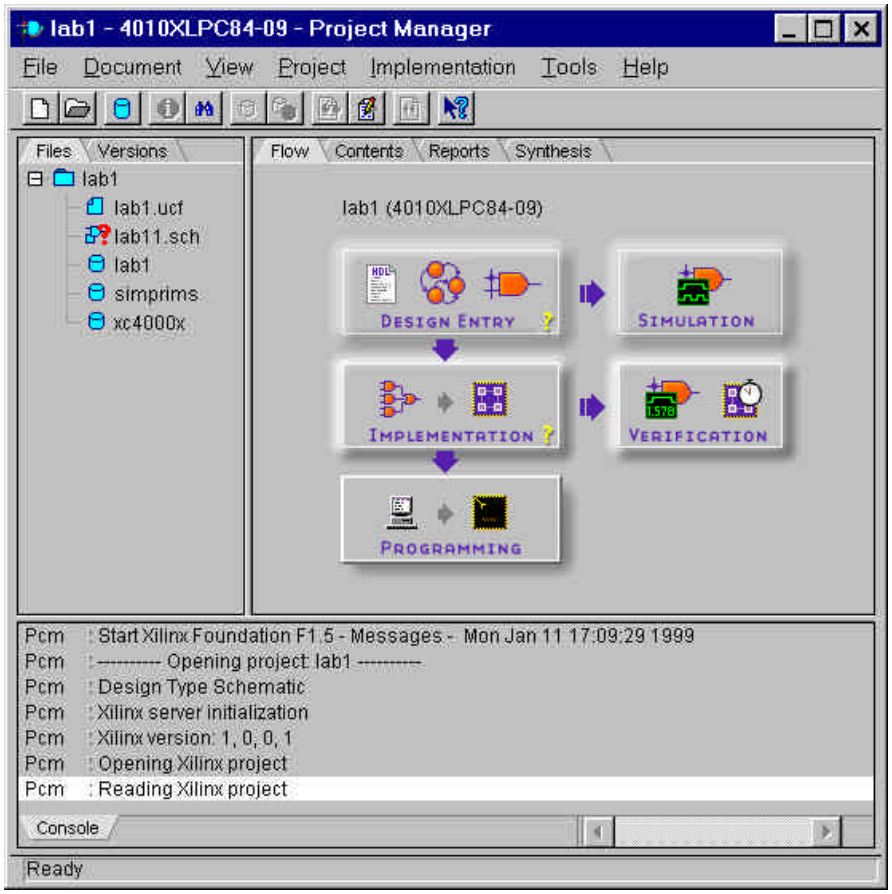
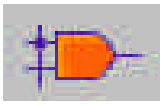


Figure 7: Xilinx Project Manager

Note that in the upper right pane, there are three icons in the "Design Entry" box. Each icon corresponds to a different method of specifying a circuit design. In this lab, we will specify the full adder circuit by drawing its schematic. Click the rightmost button in the Design Entry box to enter the Schematic Editor, where you will enter the schematic design of your full adder.



The Schematic Editor should open with a blank sheet, as shown in Figure 8. The scematic 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.

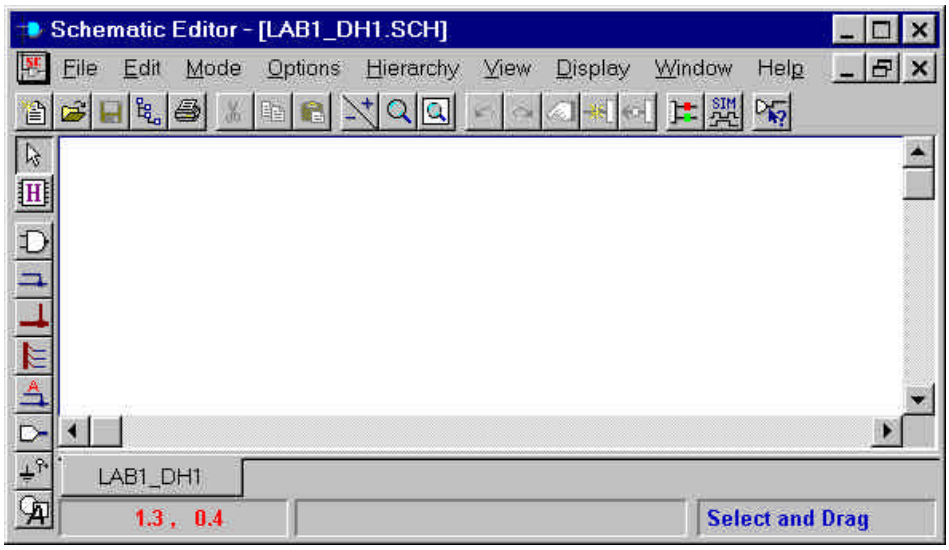


Figure 8: Xilinx Schematic Editor

First you will need to specify the inputs and outputs of the circuit you want to draw. You do this by drawing the input and output terminals directly onto the schematic sheet. First click the "Hierarchy Connector" button.

This should bring up a "Hierarchy Connector" window (Figure 9), in which you can enter the name and type of the input or output signal. Type "A" as the terminal name, select "INPUT" as the terminal type, and press OK.

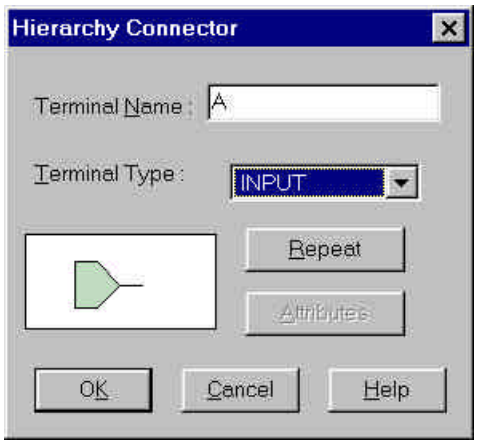
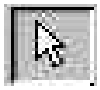


Figure 9: Hierarchy Connector Window

Your mouse cursor will now appear different, and you can click to place the input terminal onto the sheet. Do the same thing to draw the B, CARRY_IN, and SUM terminals now. Be sure to make the SUM terminal of type "OUTPUT."

Note that you can change the name or type of a terminal after creating it by double-clicking on it. Properties of other objects in the schematic can also be edited this way.

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 and Drag" tool, which changes the mouse cursor into a crosshair shape.



Also note that you can use the scrollbars at the right and bottom to scroll across the sheet, and you can zoom in or out by clicking on the "+" or "-" halves of the zoom button.

Next 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 4. To draw these gates, first click on the "Symbols Toolbox" button to open a window that lists the available circuit symbols.

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 (alternatively, you can type "XOR2" in the text box at the bottom of the Symbols window). Notice that when a symbol is selected, a short description appears in the space at the bottom of the window, as seen in Figure 10.

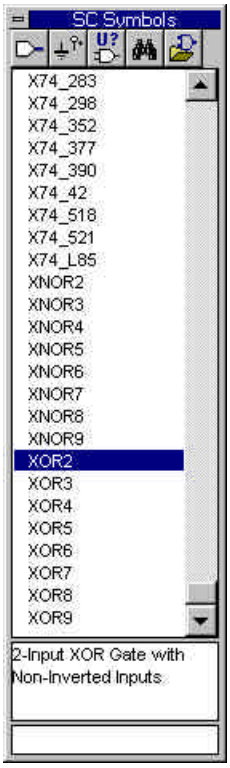


Figure 10: Circuit Symbols Window

With the "XOR2" symbol selected, drag your mouse back over the blank sheet in the schematic editor. You will see the XOR gate near the mouse pointer. Click to place the XOR gate onto the sheet. Select "XOR2" again in the symbols window to draw a second XOR gate.

You are now ready to wire up this part of the circuit. Push the "Draw Wires" button.

Wires can be drawn much the same you would draw lines in a standard drawing program. You can click once on a terminal to begin drawing a wire. While drawing a wire, you can click again to create a 90 degree bend at the current point. Clicking on a another terminal automatically connects the wire to it and ends the drawing of that wire.

The software attempts to route wires intelligently for you. If you use the "Select and Drag" tool (shown previously) to move components that have wires connected to them, the wires will stay connected. You can also use the "Select and Drag" tool to drag sections of 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 4.

You are now ready to complete your schematic of the full adder by drawing the logic for CARRY_OUT that you designed in Part 1 to complete Figure 5. 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

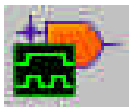
Do not add a second set of input terminals for A, B, and CARRY_IN. Instead, note that you can connect multiple wires to the same input terminals (or you can connect wires to other wires to create branches).

This completes your full adder schematic and your introduction to the Schematic Editor. You may want to print out your schematic now since you are required to turn it in. To do so, first go to File•Page Setup and choose "A – 11x8.5in" as the paper size, then select File•Print. Finally, choose File•Save to save your schematic before you exit the editor.

3. 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 use the **Logic Simulator** to test that your circuit correctly implements the full adder specified be the values in Table 1.

From the Project Manager, press the "Functional Simulation" button to start the Logic Simulator.



Pressing this button may cause the Project Manager to open a window complaining:

> "Schematic netlist lab1_xx is older than schematic. Update netlist from Schematic Editor?"

Answer YES to this dialog if it appears.[1] You will also see warnings in the Project Manager about hierarchy connectors on the top level schematic. You may ignore these warnings. You should now see the Logic Simulator window. The "Waveform Viewer" part of this window is divided into left and right parts. When performing simulation, the input and output signals are listed in the yellow colored area on the left hand side, and the values of the signals are graphed in the white area on the right hand side.

Before beginning a simulation, you first need to select which input and output signals you would like to simulate. Choose Signal•Add Signals from the menu. This should bring up a "Component Selection" window, as in Figure 11. In the "Signals Selection" pane on the left hand side of the window, double-click each of the signals A, B, CARRY_IN, CARRY_OUT, and SUM. Red check marks should appear beside each signal name as in the figure. Close the Component Selection window now (but not the Logic Simulator!).

[1] This occurs when the netlist was generated before the last time that the schematic was saved. The netlist is generated from the schematic, as is an internal representation of all the connections that you made in your schematic.

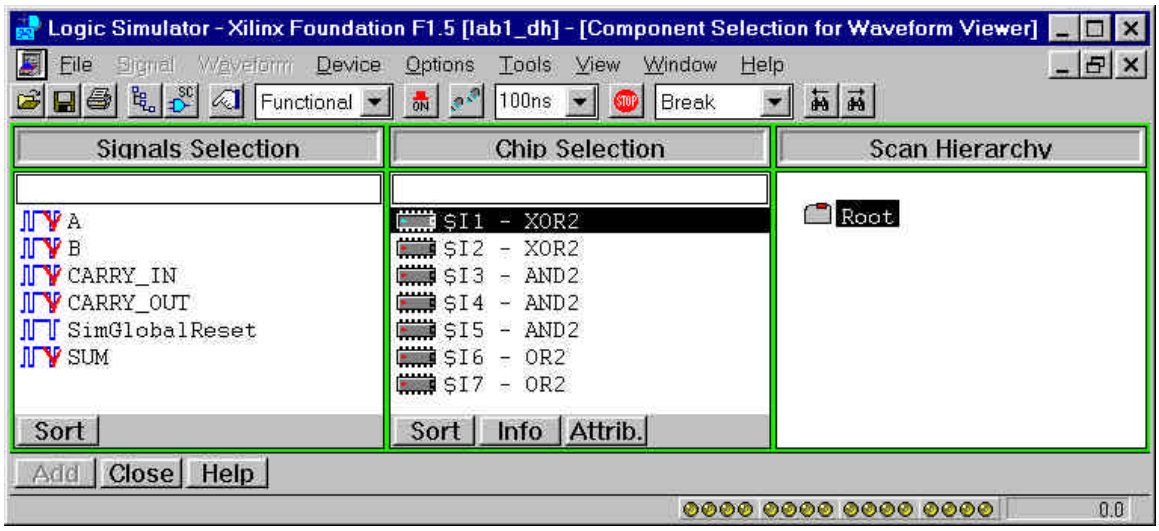


Figure 11: Selecting Signals to Simulate

The signals that you selected should now appear in the Waveform Viewer window, as seen in Figure 13. Note that in the right hand area of the window, there are now gray dotted lines. When the simulation runs, the signal values will be graphed as black lines in this area. When a signal's value is 1, it's line will lie above the corresponding dotted line. When a signals value is 0, it's line will lie below the dotted line. The horizontal axis is time.

You now need to set the amount of time for which the signals will be simulated (how far their values will be graphed). In the middle pulldown bar of the Logic Simulator window, type "40ns" to allow the simulation to run for 40 nanoseconds. This will be a sufficient amount of time to observe the full operation of your full adder.

You now need to adjust the time scale so that the waveforms will be visible when the simulation runs. This is done by clicking on the buttons that are directly above the place where the signal names appear. The button on the left looks like a fine-tooth comb, and the button on the right looks coarser.

Use these buttons to zoom in or out on the time scale until the value between the buttons reads "500ps/div."

There is one more thing you need to do before starting the simulation: you need to specify the input values for the circuit. The simulator will use these values, along with the information from your schematic, to generate the output values for the circuit. Since we want to test the complete operation of the full adder against the values in Table 1, the ideal inputs would be those in the left hand columns of the table. As mentioned before, these values are 3 bit binary number in numeric order.

The simulator provides a feature that allows you to drive the inputs of your circuit from the outputs of a binary counter. To do this, first press the "Select Stimulators" button.

This should open a "Stimulator Selection" window as in Figure 12. Ignoring the complex appearance of this window, focus on the top row of circles labeled "Bc" (they look like yellow LEDs). Each circle corresponds to one bit of a binary counter. The rightmost circle is the least significant bit of the counter. Click and drag from the rightmost circle onto the signal name for "A" in the Logic Simulator Window. A red "B0" should appear to the right of the signal name. In the same manner, drag the second least significant bit of the counter onto the signal name for "B," and drag the third least significant bit onto "CARRY_IN."

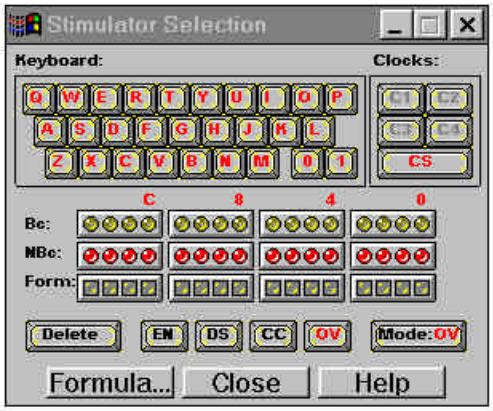


Figure 12: Stimulator Selection Window

The inputs of your circuit are now connected to a binary counter, and the input values will be exactly those specified in the left-hand columns of Table 1. Since the binary counter spends 5 nanoseconds on each number, the simulation should generate the outputs that match the values shown in the right-hand columns of the table, spending 5ns in each row.

At this point, your Logic Simulator window should appear similar to the one in Figure 13. Note that bits B0, B1, and B2 of the binary counter are connected to the inputs, as signified by the red text just after each input's name.

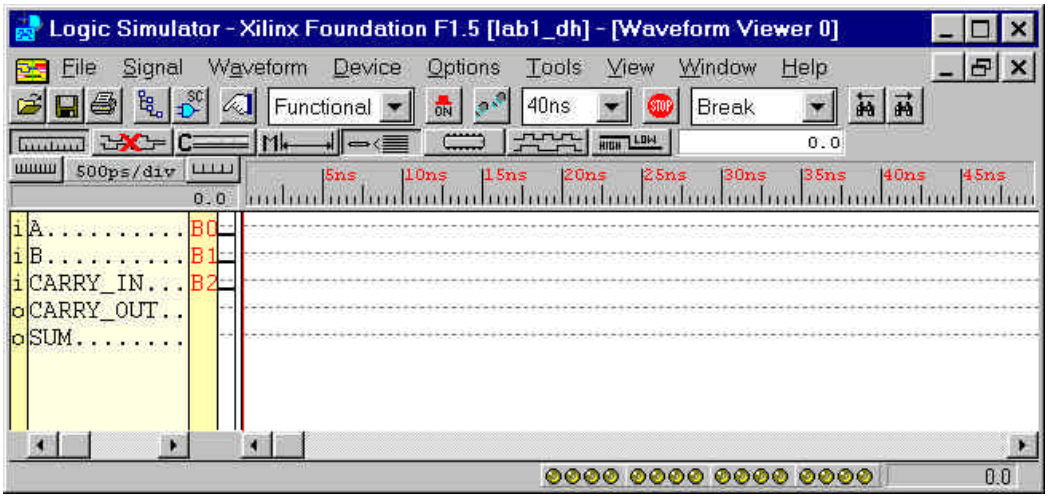


Figure 13: Xilinx Logic Simulator

You are ready to run the simulation of your full adder. Run the simulator for one timestep (40ns) by pressing the "Simulation Step" button now.

The Waveform Viewer should now display a graph of the input and output signals. Verify that the output signals match those in Table 1. If not, try to find and correct the bug in your design, and update your schematic and run the simulation again.

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. In the print dialog box, change the Print options Header field to something meaningful like your name and the name of the lab. Congratulations on your completion of lab 1!

What to Turn In

You must provide a hard copy of each of the following items, by which your grade for this lab will be determined:

1. Please indicate how many hours you spent on this lab. This will not affect your grade, but will be helpful for calibrating the workload for next semester's labs.
2. Your completed truth table, including the values in the CARRY_OUT column. A hand-written copy is fine.
3. A printout of your completed schematic, including the logic gates for both SUM and CARRY_OUT. This can be produced using the File•Print•Current Sheet feature of the Schematic Editor.
4. A printout of your simulation of the full adder, including all inputs and outputs. This can be produced using the File•Print feature of the Logic Simulator.