EXPERIMENT NUMBER 7
HIERARCHICAL DESIGN OF
A FOUR BIT ADDER (EDA-2)

Purpose

The purpose of this exercise is to explore more advanced features of schematic based design. In particular you will go through the hierarchical design of a four bit adder.

References

Tutorial: Quartus II and ModelSim

Preliminary

Read through the tutorial to familiarize yourself with the steps you will be performing. Construct a truth table for the circuit in Figure 7-2 of the tutorial and verify to yourself that it represents a full adder circuit.

Procedure

Work through the tutorial in the lab. Take notes as you are going through the procedures. I can guarantee you'll be coming back to this chapter for help later. Answer the questions below and submit them together with hardcopies of the schematic along with any other material your TA might require.

Questions

1. Why is it not important which signal goes to which input of a full adder?
2. Actually there is a slight difference between full adder inputs. What is the difference between the A, B, and C inputs shown in Figure 7-2 of the tutorial?
3. What is a symbol?
4. What is the purpose of a bus?
5. The tutorial led you through bottom up design. Briefly describe the procedure you would use for top down design.

CpE 112 Tutorial
More Quartus II

Purpose

The purpose of this exercise is to introduce you to *hierarchical design* and more advanced functions in Quartus II. You will design the symbol and schematic for a full adder component and then instantiate that component in another schematic in order to implement a four bit ripple carry adder. You will also be introduced to buses which are like subscripted variables in a programming language.

The kind of hierarchical design we'll explore in this tutorial is sometimes called *bottom up design* since we'll start with the design of a low level component and then use that in a higher level design. We will use a symbol for the low level design instead of creating multiple copies of its schematic in order to reduce the complexity of the higher level design. Schematics which consist of symbols which represent other lower level schematics are often called hierarchical schematics or designs as opposed to flat schematics which consist entirely of primitive components such as gates and flip flops. We could have chosen to use a *top down* design methodology by creating a top level symbol and then creating the underlying schematic instead of bottom up design where we start with schematics and then create symbols but we'll do it this way this time and save top down for another day (and a more complex design).

When you are finished with this exercise you should have a solid understanding of the basics of schematic capture and logic simulation with ModelSim. While you won't be an expert you'll be well on your way toward becoming one and should be able to explore the basics and more advanced features of FPGA design tools in depth on your own using the online reference and user's manuals.

Getting Started

After creating a new project named lab7 in Quartus II, the first task will be to design a full adder component for the four bit adder. In the Quartus II display select File |New. Since we want to illustrate the schematic-entry approach in this section, choose **Block Diagram/Schematic File** and click OK.

A logic diagram for a full adder is shown in Figure 8-1. You should convince yourself that functions S and CY do in fact implement the Sum and Carryout functions of a full adder. S is straightforward and is high (1) whenever there are an odd number of input lines high. CY is not quite as obvious but is high whenever A and B are high or C and one of A or B is high. This realizes the function `AB+CAB'+CBA'` which is the carry out function of a full adder.

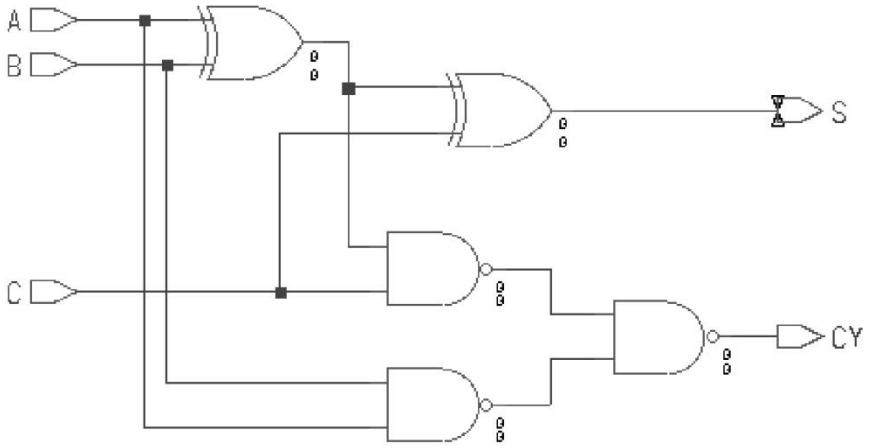

Figure 8-1: Full Adder Schematic

Save this file with the name fulladd. Once getting successful compilation, go to File, click Create/Update and Create Symbol Files for current file as shown in Figure 8-2. This symbol will be used in the four bit adder.

Four Bit Adder Design

Create a new Block Diagram/Schematic File. To import the schematic symbols, double-click on the Block Editor screen, or select **Edit | Insert Symbol**. This command opens the window in Figure 8-3. Click on the + next to the label Project on the top-left of the figure, and then click on the item *fulladd* to select this symbol. Click OK to import the symbol into the schematic.

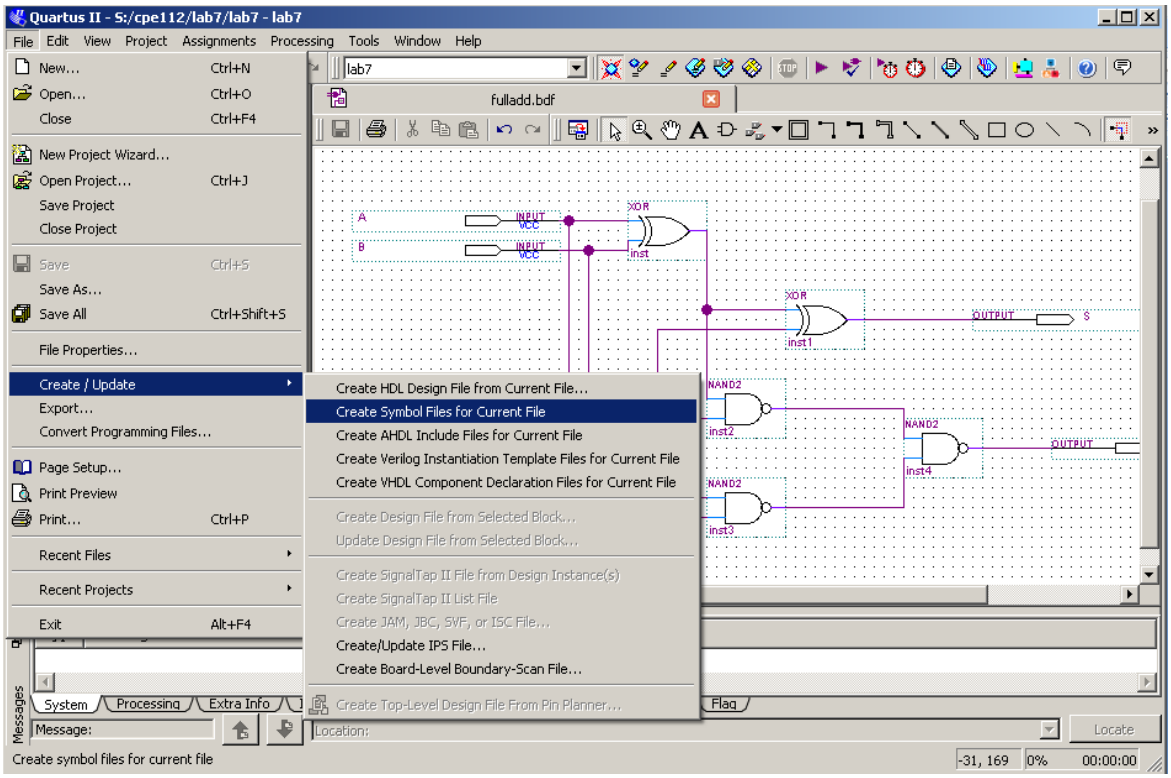

Figure 8-2.

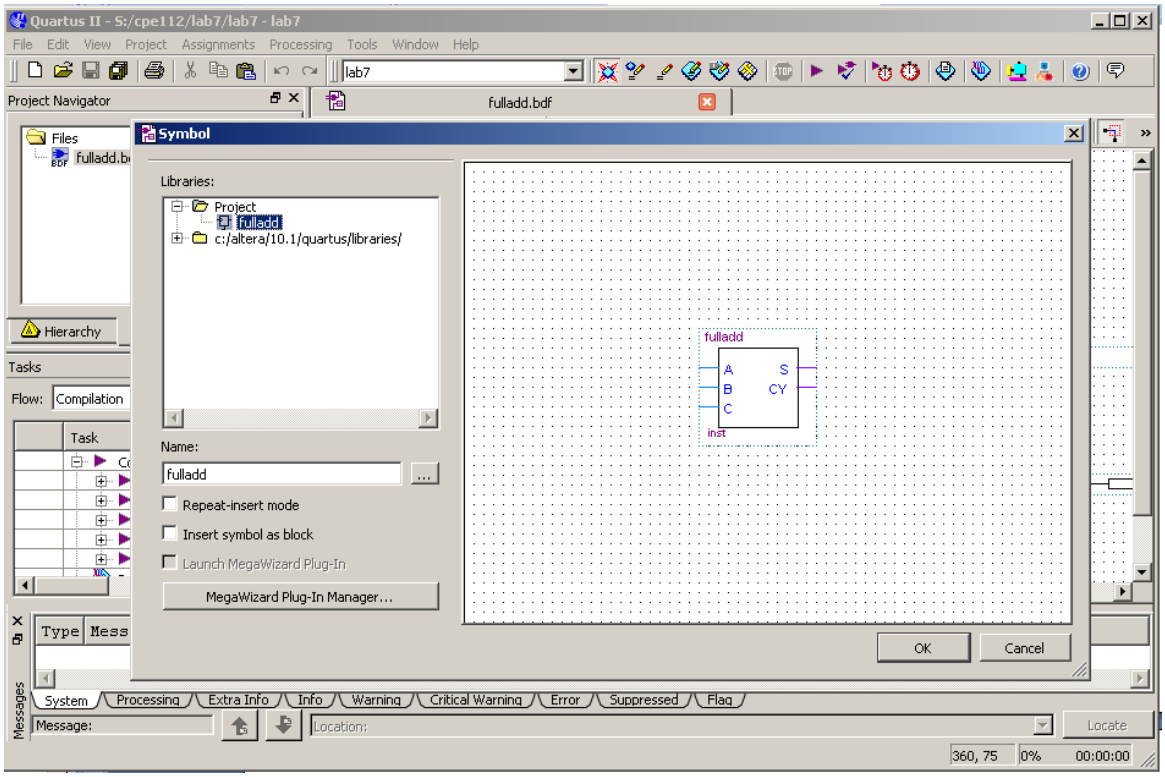

Figure 8-3.

Buses are nothing more than nets with a special naming syntax and correspond physically to a bundle of wires or like a subscripted variable in a programming language like C or Fortran. Normally buses are shown on a schematic as a line segment that is a little fatter than a normal net but that is purely for visual purposes. In general all buses must have a name. The system must be told somehow that a particular net corresponds to more than one wire and the name is the only way we have for doing that at the moment. The orthogonal and diagonal bus tool can be found at the top of the schematic screen.

Add a portin to each bus and name them A[3..0] and B[3..0]. The bus naming convention consists of a netname (A or B) and a subscript range in parentheses. By convention the subscript on the right is the LSB so that if for example we assign the bus A[3..0] it means that A[3] is high and bits A[2..0] are low.

Also, notice that each wire connected into a bus has to be assigned a name such as A[3], A[2], A[1] and A[0] . The same thing needs to be done with the output S[3..0]. Finally, the circuit is shown in Figure 8-4.

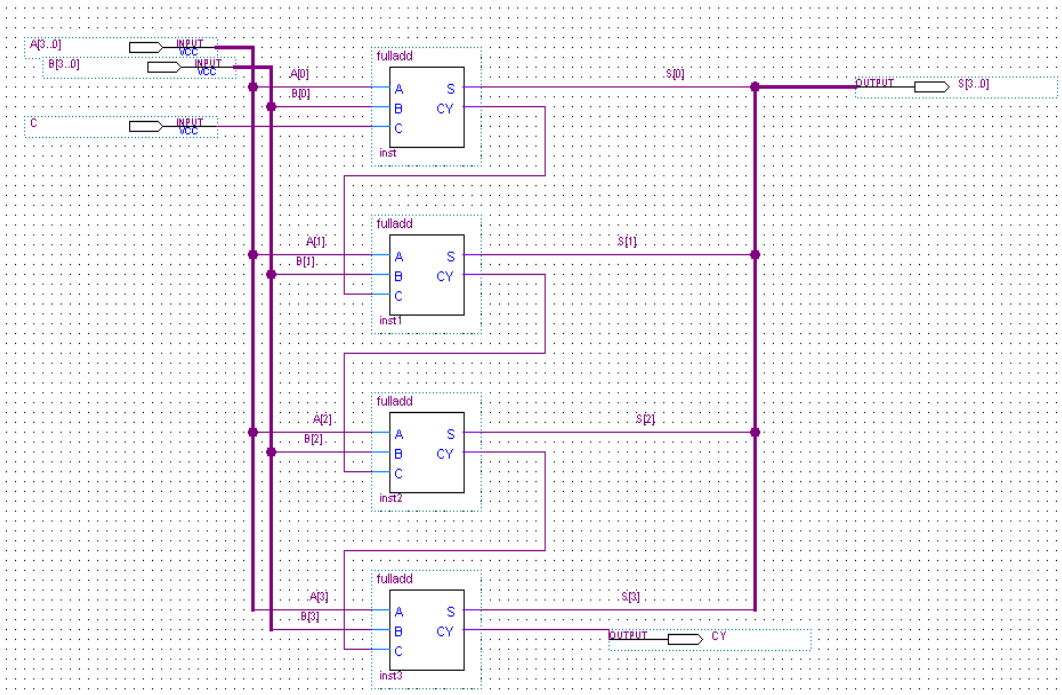

Figure 8-4.

Save this file with the name lab7. Once getting successful compilation, go to File, click Create/Update and Create HDL Design file from current file for both lab7.bsf and fulladd.bsf. The two VHDL file are lab7.vhd and fulladd.vhd.

Simulation

Start the Modelsim software. Create a new project named lab7. And add both lab7.vhd and fulladd.vhd as the existing files into the project. After the compilation of all files, we start simulating this design by generating waveforms to be applied to the CIN, A, and B signals. Since both A and B are four bit signals, we want to exercise all 512 combinations of 9 (4+4+1) bits and we'll let A cycle through its 16 combinations, then change B and so on. After we've cycled through 256 combinations of A and B we'll set CIN to one and repeat it all again for a total of 512 events.

In the wave window, left click on the "+" of input "a", then right click on "(0)" and select clock. After setting the period to be 100 and first edge to be Falling, click on OK. Repeat this procedure for the inputs "(1)", "(2)" and "(3)" with setting the periods to be 200, 400 and 800 respectively. Then do the same thing for input signal "b", set the clock periods for "(0)", "(1)", "(2)" and "(3)" to be 1600, 3200, 6400, and 12800. Lastly, select CIN and set its clock period to be 25600.

Then type **run 51200 ps,** you should be able to see the simulation waveform. Also, if you want to view the hexadecimal format of the input and output, select A, B and S, and click on format/radix/ hexadecimal.