

VALPARAISO UNIVERSITY
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT

ECE 221

Design Project #3 - 4-Bit Ripple Adder/Subtractor

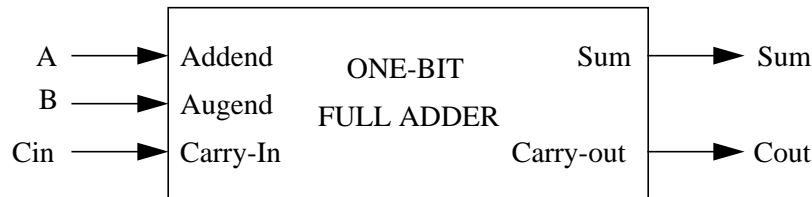
Fall 2003

Introduction: In the previous projects we have investigated the use of three functionally complete sets of gate operations for the realization of combinational logic functions: 1) AND, OR, and NOT gates, 2) NAND gates, and 3) NOR gates.

This design project focuses on two concepts: the use of complex circuits (multiplexers and decoders) to implement logic functions, and iterative design practices to create larger designs. You will use these concepts to develop a 2's complement 4-bit adder/subtractor circuit.

SECTION I. Conceptual Design - Full Adder Plus

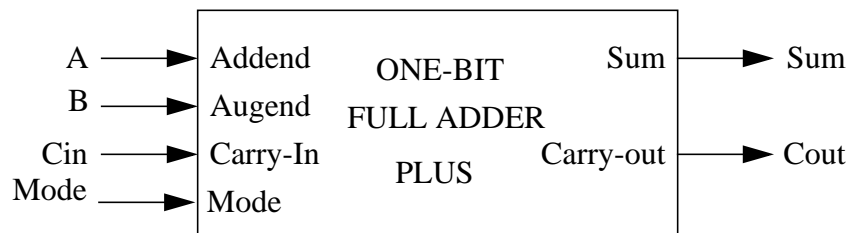
The main component of the adder/subtractor is a full adder. As shown in the diagram below (Figure 5.45 on page 151 in Dewey), a one-bit full adder has three one-bit inputs (addend **A**, augend **B**, and carry-in **Cin**), and two one-bit outputs (carry-out **Cout** and **Sum**).



One possible gate-level design for a full adder can be found on page 151 in Dewey. This design was developed to utilize XOR gates. A number of other implementations of the full adder are possible as we will see in this project.

1. Write a truth table for the one-bit full adder, with columns in the order **A**, **B**, **Cin**, **Cout**, and **Sum**. The input values for the eight rows should be in a binary counting sequence. For inputs to the truth table, make bit **A** the most significant bit (MSB) and **Cin** the least significant bit (LSB).
2. Write **Sum** and **Cout** as minimized SOP expressions.
3. Design a circuit on paper to implement the **Sum** circuit of the full adder using only four 2-to-1 multiplexers (you can select any of the signals for your select lines). Use your truth table and/or K-map to determine what to connect to the inputs of each multiplexer. Assume all inputs into the circuit are uncomplemented.
4. From inspection of your truth table for the full adder, determine (and document on paper) the appropriate connections for the implementation of the **Cout** circuit using one 3-to-8 decoder and an OR gate.

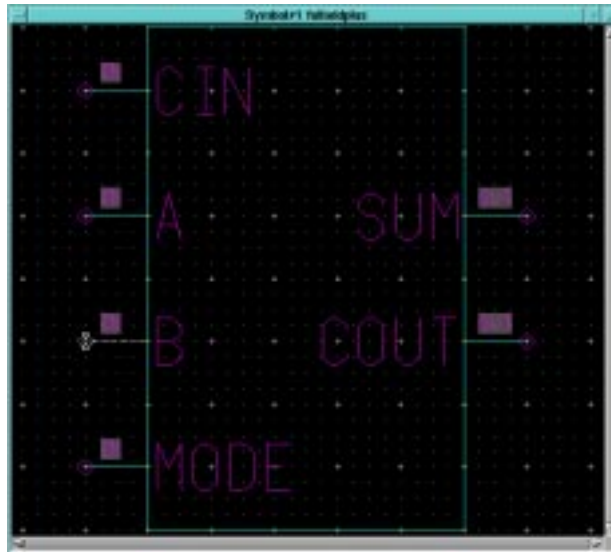
To make the full adder into a component that can either add or subtract, you need to modify your full adder to include a mode bit which will determine whether an addition (Mode=0) or subtraction is to be done (Mode=1). The final block diagram is shown below.



5. Add the appropriate logic to your full adder circuit so that your adder can either add or subtract. Remember to subtract two 2's complement numbers you can convert the second number to its negative equivalent and then add them. This means that the input B going into the full adder will be either B (if adding) or the complement of B (if subtracting). Adding the extra '1' will come later when the entire system is put together.

SECTION II. Full Adder Plus Design and Verification

1. Open Design Architect and open a new design sheet. Call the design **fulladdplus**. Add a border to your design and change the appropriate fields.
2. Create the design schematic using your conceptual design. For all of your components, use parts out of the generic library (**gen_lib**) instead of the board process libraries. If you need a +5 volts or ground component you can use **vcc** and **ground** out of the generic library.
3. Once you have created your design, you need to create a symbol for your adder that you can use in the 4-bit adder implementation. To create a symbol, select the **Generate Symbol** option under the **Miscellaneous** menu. When the **Generate Symbol** form appears, simply press **ok**. This operation invokes the symbol editor which allows you to create and modify symbols for various components. You should see a box with the inputs and outputs of your design. Move the lines and pins (the diagonals at the end of the lines) so that the order of your inputs and outputs matches the symbol shown below. When you move the line and pin, the label within the box will also move.



Once you have put the inputs and outputs in the proper order, check the symbol using the **With Defaults** option under the **Check** menu. You may get a few warnings (similar to “the following pins on the symbol are not on the interface”) which you can ignore. Finally save the symbol using the **Save Symbol** option under the **File** menu. You can then exit the symbol editor.

4. Back at the full adder schematic, check your design and then save it.
5. Use QuicksimII to verify your circuit is correct. Create a force file to check all possible combinations of inputs to verify your design is working properly. Arrange the waveforms in the Trace window in the following order, from top to bottom: **MODE, A, B, CIN, SUM, COUT**.

SECTION III. 4-Bit Adder/Subtractor Design and Verification

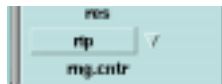
Now that you have the full adder plus working, the next step is to use that component to create a 4-bit

ripple adder/subtractor. From the in-class project you should have a schematic created for how the full adder could be connected to create a 4-bit ripple adder. Using those notes as a guide, create and verify a 4-bit ripple adder/subtractor design schematic as outlined below.

1. Open a new design sheet. Call the design **fourbitaddsub**. Add a border to your design and change the appropriate fields.
2. Your 4-bit adder/subtractor should contain only one component: the full adder plus you just created. That component is not in a library but in your directory structure. To access the component, use the **Choose Symbol** option located on the palette. When you select that option, a form will pop up. Find the schematic icon for your full adder design, select it and hit **OK**. An example of a selected icon is shown below. You will also need ports to connect your inputs and outputs.



3. Wire the full adder components together in accordance with the design you created. For the inputs and output you will be using a new item called a **bus**. A bus represents a set of nets grouped together and is used to help clarify schematics involving larger designs. In this project three buses will be used. Two for the 4-bit inputs, A and B, and one for the 4-bit output, SUM. When using both busses and single nets, you will need another device for the generic library called a “ripper.” A ripper is used to attach or detach single nets from the bus. In the **gen_lib**, find the **rip** component. Beside the component is an upside down arrow (as shown below). When you click on it, a list of rippers appears.



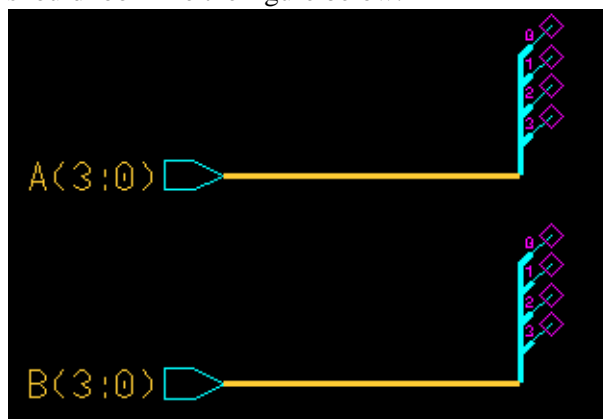
Select the **4x1** ripper and add three of them to your design. A bus is connected to the fat end of the ripper and individual nets are connected to the skinny ends of the ripper. Draw a bus from two of the rippers out to the input ports as shown below. Drawing a bus is similar to drawing a wire except you

use the **Add Bus Bundle** option on the palette instead of the Add Wire. A bus appears thicker than a wire in the schematic.



When using busses and rippers you must identify the size of the bus and tell Mentor which bits should be ripped off. In this example, the size of the bus can be identified through the input port label. Change the two input port names to **A(3:0)** and **B(3:0)**. This gives each bus a width of four.

Next, the property associated with each ripper must be changed from an 'R' to a number corresponding to the appropriate bit on the bus. To change the properties quickly, select the **Text** option from the schematic palette. Then select the **SEQUENCE TEXT** button. This will pull up another window in which you just need to hit **OK**. Then holding your **LMB** down, draw a box around the ripper (from top to bottom). The **R** values should change to increasing numbers. If you make a mistake, you can always use the **CHANGE VALUE** button to change any property. Your final two input ports and rippers should look like the figure below.



The final step is to connect the nets from the proper ripper number to the individual full adders. Again, use your class notes as a guide.

Follow the same procedure for adding the output bus. To orient the output ripper properly, select it and use the **Flip** option on the palette. Label the output port **ANS(3:0)**. For the single inputs and outputs use the names **COUT** and **MODE**.

4. Once you have connected all of the components via busses or nets, check your design. If you don't have any errors save the design.
5. Use QuicksimII to verify your circuit is correct. Arrange the waveforms in the Trace window in the following order, from top to bottom: **MODE**, **A(3:0)**, **B(3:0)**, **ANS(3:0)**, **COUT**. Using previous forces files and **adder.forces** in the ECE221 public directory as a guide, create your own force file to verify your circuit is working correctly. Test the following cases: (3+4), (7-3), (-5-(-7)), (3-6), (5+6), (-8-3) Run the simulation and verify that your design is working properly. Your bus waveforms will show values in hexadecimal format so you may need to convert them to binary in order to check if the sum/difference is correct. On the waveform print-out write the binary equivalents for each test case

and the associated output right above where the test is being done. Also, write on the waveforms if there is an overflow or underflow occurring for that test case. If you find a mistake you do not have to exit QuickSim. You can correct the error in Design Architect and then use the **RELOAD MODEL** under the **DESIGN CHG** palette. You can also zoom in and out of the trace window using the command under the **View** menu.

SECTION IV. WHAT TO TURN IN:

1. Design project header page containing your name and the signed honor code.
2. Your conceptual design information placed a blank piece of paper (answers to questions 1-5).
3. A copy of your design schematic and the output waveforms for the full adder plus and the 4-bit ripple adder/subtractor.

ECE 221 Digital Logic Design
Design Project #3: 4-Bit Adder/Subtractor

October 21, 2003 (lab time)

Name: _____

Honor Code Pledge:

Signature: _____

Please staple this sheet to the front of your assignment