CSE 370 Introductory Laboratory Assignment

Making a Full Adder with a Multiplexer, Decoder, and FPGA (Field Programmable Gate Array)

Assigned: Monday, October 17, 2008
Due: End of Lab Section

Objectives

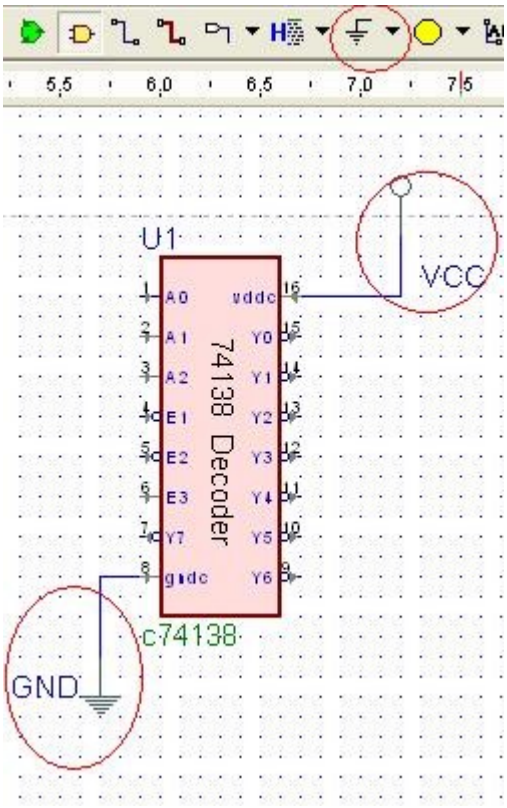
In this laboratory assignment you will learn to use multiplexers, decoders, and FPGAs to create a full adder. If you aren't familiar with them you can refer to Chapter 4 of Contemporary Logic Design. We will provide you with a brief summary of FPGAs are and what they can do in this lab, however you can also refer to the wikipedia page for more information.

In this lab you will design a full adder circuit, first using a decoder, then a multiplexer, and finally on an FPGA.

Before You Begin

1. There are a few points you should keep in mind for this lab. First off, the four bit adder/substractor circuit that you will be using in the FPGA tutorial should be designed with the provided gates in Active-HDL (the yellow gates). You DO NOT use the gates in lib370 because they will not synthesize at all and your circuit will not work right. If you designed your four-bit adder/subtractor with the lib370 gates, you are going to have to replace them with the gates in the built in library.
2. The FPGA is much more flexible, with 18000+ plus logic elements you can basically design anything to your hearts content. However up to this point you have only been using switches and LEDs, starting from this lab you will have to control the LEDs and switches yourself instead of directly connecting up wires. This means you will be able to avoid most wiring problems, however there is a learning curve involved with the FPGA that is why we are presenting you with this lab to familiarize yourself with the FPGA.
3. You should have your verilog implementation/design of a full adder from Lab 3 and the verilog full adder test fixture (FA_tf.v).

Remember when you are designing your decoder and multiplexer design of the full adder you only use the C74 chip packages included in the lib370 library. An example is provided below.



Notice that the chips in the cse370 library need to be powered just like real world chips. That is why there is a VCC connection and a GND connection in the picture. Be sure you connect up the right ports! You can always look on your chip maps if you are unsure about anything.

*****NOTE**
If you look on the chip shown above you will see a few inputs called E1, E2, and E3. These are the enables and if they are not set to the correct logic level the chip will not work. E1 and E2 are shown with circles on their inputs. This means they are active low and they must be set to GND in order for the chip to be enabled. E3 does not have a circle and must therefore be set to VCC. ALL enables must be hooked up correctly in order for the chip to work. Note that in the chip maps E1 and E2 are shown with a line over them. This is another way of indicating that they are active low.

Tasks

Part 1: Can be done outside of Lab Time

1. Your first task is to implement a full adder with a 3:8 decoder (‘138 chip) as your primary piece of logic. When you design your full adder in Active-HDL don’t forget to use the C74 chips provided for you in the lib370 library. It will make building your designs a lot easier later on. Don’t forget to apply what you’ve learned in class such as deMorgan’s law to pick the right sets of chips. Also don’t forget to set the correct enables on the chips in the design. Refer back to the part in the lab about enables if you skipped it. Also notice that all of the outputs are active low. This means that at any given instant only one output will be 0 and the rest will be 1.

   When you think you have provided a working design test it with the provided test fixture from

Lab 3. It is linked here: FA_tf.v. You don't have to hand in your test diagram, but we do need to see it. Call over a TA so they can check you off for this first part. Remember to check to make sure your test fixture actually passed with the right results. If you see X's you did something wrong.

2. Your second task is to implement a full adder with two 4:1 multiplexer ('153 chip) for each output that you need to compute (sum and carry-out are the two outputs you need to compute). You can use any other necessary chips such as inverters to implement your full-adder. To save you some wiring you should see that the ('153) contains two 4:1 multiplexers so you should only need to have one 153 chip placed on your prototype board. As a reminder again, use the C74153 chip in the 370lib when you design it in Active-HDL. You should not be using the included multiplexers in Active-HDL.

S1	S2	Z
0	0	I0
0	1	I1
1	0	I2
1	1	I3

A 4:1 multiplexer with inputs S0, S1, I0, I1, I2, and I3

Note that hooking up I0, I1, I2, and I3 to VCC or GND is an easy way to implement any 2 variable truth table however a full adder is a 3 variable function, therefore you will have to be a little bit more clever about what you hook I0-I3 up to.

When you think you have provided a working design test it with the provided test fixture from Lab 3. It is linked here: FA_tf.v. Once again, if it is working and you don't have strange outputs like X's, you should call over a TA so that they can check off your design.

Part 2: Needs to be done during Lab Time

3. Complete the entire FPGA Tutorial that explains how to compile a circuit into a FPGA using Active-HDL and the Quartus compiler. **Hold onto the programming file, you made in the tutorial. You will need it to get your lab checked off**. Once you have completed the tutorial you should compile your full-adder Verilog module from your previous lab and program it onto your FPGA following the tutorial; This will make sure you understand how to program the FPGA without following a step by step guide. Test out your FPGA and make sure it behaves like a full adder.
4. Now wire up your multiplexer implementation of your full adder using switches for A, B, and Ci and LEDs for Sum and Cout. Once your multiplexer implementation is working AND you have completed the FPGA tutorial and have a programming file call over a TA to be checked off. Note that both implementations (FPGA and multiplexer) will not work at the same time because programming the FPGA overrides the usual functionality of the LEDS. In other words they will no longer turn on when the corresponding input on the board is being driven high; they will only turn on when the corresponding pin on the FPGA is set high (see pinouts.html. Once the board is reset the multiplexer implementation should work again.

Lab Demonstration/Turn-In Requirements

A TA needs to "Check You Off" for each of the tasks listed below.

1. Show your decoder implementation using c74XXX gates in Active-HDL to a TA.
2. Show your multiplexer implementation using c74XXX gates in Active-HDL to a TA.
3. Demonstrate the 4-bit adder FPGA implementation that you made with the FPGA tutorial, and the multiplexer implementation built with chips on your prototyping board.

Comments to: cse370- webmaster@cs.washington.edu