CSE 370 Introductory Laboratory Assignment

Designs in FPGAs (Field Programmable Gate Array)

Assigned: Friday, October 24, 2008
Due: Beginning of Next Lab Section

Objectives

In this laboratory assignment you will work extensively with Verilog and Active-HDL to design an 8-bit accumulator. When you have finished designing the accumulator you will implement it on your FPGA to try it out. To help you through this process we will guide you with design steps for each part of the accumulator; however it will be your job to figure out how to put them together properly and make it work on the FPGA. If you finish early, feel free to explore the FPGA, you have over 18000 logic elements in the FPGA, more than you will probably ever need for what we're doing.

Before You Begin

1. For this lab, it is strongly suggested that you create a NEW design in your workspace. Call it: lab5. Typically we don't enforce this kind of organization, however good organization makes debugging easier and it reduces the chance for errors.
2. You will need the pinouts in order to write the (.qsf) file for your designs. You can access the link here: Pinouts.

Important Verilog to Know:

In this lab you will be coding a portion of your design in Verilog, a hardware description language. Although it may look like a programming language Verilog is actually a hardware language which acts as a medium in which to describe hardware with words. An important thing to remember about Verilog is that your code runs in parallel! It is not sequential like most programming languages, this means that any statements you write will all be run at the exact same time. **Sequential logic does not apply to the previous statement.**

In this lab you will have to assign a 7-bit value to a 7-bit bus for your hexadecimal display. There are many ways to do assignments in Verilog, although there are two basic ones which you should know about. One way is to directly assign the decimal value: "out = 48;", another way is to directly assign the binary value: "out = 7'b0110000;". Both of these methods work, it's up to you on how you want to assign your values. Remember, if you choose to do the binary format, the rightmost digit is the least significant and the leftmost digit is the most significant.

In this lab you will also have to be able to do comparisons and assign values based on the comparisons.

There are two ways to do this, and based on the situation one may be better than another. One method is to use the ternary where you do a comparison and based on the comparison either one statement is executed or another. For example "out = (X== 1) ? 48 : 0;". In this example you see that 'out' will be assigned the value 48 if X is equal to 1, otherwise it is assigned the value 0. You can nest these ternaries inside each other as well, which gives you many layers of comparisons.

Another way to do comparisons is inside an ALWAYS block in Verilog. Inside ALWAYS blocks you can do if statements and case statements. However, an ALWAYS block is only triggered when the variable attached to it changes. Take a look at the example Verilog code here: example.v. It will explain the usage of an ALWAYS block for comparisons. **Read the comments in example.v!**

Hexadecimal Display Background Information:

We will begin by writing a decoder to display values to the hexadecimal display on the board. If you look to the left side of the board above LEDR5-9 you should see 4 hexadecimal displays. They are labeled HEX0, HEX1, HEX2, and HEX3. They are called HEX because they are made to display the hexadecimal values 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b, C, d, E, F. If you recall, hexadecimal values are made up of 4 binary values. So for example, the binary value 0001 would correspond with the hexadecimal value of 1, and the binary value 1111 would correspond with the hexadecimal value of F.

With this background information you need to build a hexadecimal decoder that takes in 4 bits and outputs a 7 bit value for the 7-segment hexadecimal display. It is a 7-segment display because each of the 7 segments in one hexadecimal display can be toggled on and off. When you give it a high value (1) it turns off, and when you give it a low value (0) it turns on. Finally you need to know how the 7 segment display is ordered, you can go to documentation here: Documentation and look at page 30 and 31. There will be a nice picture that tells you how the 7-segment display is ordered.

Tasks

1. Your first task is to code up the 4-bit number to 7-bit segment display. Make sure your Verilog module has a 4 bit input bus and a 7 bit output bus. Once you have made your Verilog module you can test it by writing a (.qsf) file which directs the 4 inputs to SW0-3, and the 7 ouputs to HEX0, 0-6. Writing the (.qsf) for a Verilog file is just like writing a (.qsf) for a Block Diagram file (.bde), you have to assign pins to input and outputs. So in the case where you have the following at the beginning of your Verilog file:

   ```
   module toHex (ouput [6:0] y, input [3:0] x);
   ```

   You will need to assign pins for in[0], in[1], in[2], in[3], out[0], out[1], out[2], and out[3], etc. If you don't understand this concept ask your TA to come over and explain it to you in more detail. After writing up your (.qsf) you should implement your design onto your FPGA you can test your 4 bit to HEX decoder to make sure it works. Go through all the values 1-15 and make sure the HEX displays the correct value. If you are too lazy to program it onto the FPGA, verify your code in simulation.
2. Your next task is to download the accumulator we have provided here for you: register_8_bit.v. What this accumulator will do is save up to 8 bits inside itself in the form of registers. Each time you add or subtract it will add or subtract the value you specified from the saved value inside the accumulator. Negative numbers are stored but your 7-bit segment display will only show

HEX so it will be difficult to tell. Now you should test your accumulator to figure out how it works. Create a block diagram and add the accumulator and four 4-bit to hex displays onto the block diagram. Create the following inputs and outputs on your block diagram:

1. One, 8-bit input bus that will be driven by SW0-7.
2. Two, 1-bit wire that will be driven by KEY0-1 which will be clear and store.
3. One, 1-bit wire that will be driven by SW9 which will be addsub. (Which we will use later for your 8 bit adder/subtractor)
4. Four, 7-bit outputs that will be driven by HEX0-3 will be for the outputs of your 4-bit to hex converter.

You should hook up the 8-bit input that is driven by the switches to the 8-bit input on the accumulator. Also hook up the clear and store signals on the accumulator to the two inputs that will be driven by KEY0-1. Now add four copies of your 4-bit to hex converters to the block diagram. The first two will be connected to the 8-bit input of the accumulator, and the last two will be connected to the 8-bit output of the accumulator. Now you have four 4-bit to hex converters with 7-bit outputs. Hook them up to the four 7-bit outputs you made earlier.

Now you need to write a pin assignment file (.qsf) for all of your inputs and outputs. When that is completed, synthesize and implement your design onto the FPGA. Test to make sure that it is working correctly. Does the value of the switches display on the HEX display? Does the value saved inside the accumulator show on the HEX display as well? When you feel it is all working call over your TA for a check off.

3. Your next task is to create an 8-bit adder/subtractor. In short, an 8 bit adder/subtractor is a module which takes in 8 bits for A, 8 bits for B, and a single signal for if you want to add or subtract and returns an 8 bit Sum. Write this 8-bit adder/subtractor up in Verilog. Remember that Verilog has built in addition and subtraction operators so you should make use of them.
4. When this part of your design is completed you need to bring it all together. How can you use the accumulator and 8-bit adder/subtractor to add together the value on the switch and the value saved inside the accumulator and save it back to the accumulator? When you figure it out wire it up correctly and write the (.qsf) file for your design. Synthesize and Implement it onto the FPGA and test it out. Does it work? If it does call over your TA for a checkoff.

Lab Demonstration/Turn-In Requirements

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

1. Check off your working Hex Decoder connected to the accumulator.
2. Check off your working 8-bit accumulator.

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