EEL 3705L, Digital Logic Design Lab, Spring 2011

Lab Assignment #5: Design a Hex Adding Machine Using Mixed VHDL & Schematics

Version history:
v1.0, 2/23/11: Created the initial version of the assignment. –M. Frank

1. Document description:

This document describes the fifth full lab assignment (and the 1st lab assignment after the midterm) for the Spring 2011 semester of EEL 3705L, Digital Logic Design Laboratory; this assignment's lab periods will take place the week of Mon. Feb. 28-Thu. Mar. 3rd.

2. Assignment Synopsis:

In this assignment, students will create a modular logic design using both VHDL & schematic modules, including custom flip-flops, to implement a sequential adding machine for unsigned bytes displayed as pairs of hexadecimal digits. This assignment is to be carried out as a structured design process. Students should design and test their circuits at home in the simulator, and then test their designs in lab on the boards.

3. Educational Objectives:

This assignment is intended to:

1. Give students practice at VHDL coding of simple combinational logic modules.
2. Exercise students' understanding of fundamental concepts of sequential logic, by making them build a working 8-bit clocked data register from scratch (from basic logic gates) in a modular design.
3. Give students practice at wiring up, simulating, and debugging modular sequential designs using the Quartus tool.
4. Continue exercising students' ability to go through a structured design process with all appropriate documentation.

4. Design Objective:

In this assignment, your task is to design and prototype a simple sequential logic circuit, in a modular design style, to implement a simple one-byte hexadecimal adding machine, which accepts an 8-bit binary input, and increments or decrements a one-byte accumulator by the input value each time a button (call it 'enter') is pushed. The input value and the current accumulator value should each be displayed in unsigned hexadecimal.

Also, your design must meet the following additional requirements, which are included for pedagogical purposes:

1. To give you practice coding in VHDL, as part of your design you must code a hex byte display module in VHDL. This module should accept an 8-bit binary value as input and produce two 7-bit output vectors, for driving the left and right

symbol blocks in a pair of adjacent symbol blocks on a 7-segment display, as appropriate to display the unsigned byte value in hexadecimal.

2. To exercise your understanding of the basic elements of sequential logic, your design must include an 8-bit register that is implemented *from scratch* (out of basic logic gates) in a modular design style, using custom schematics *only* (no VHDL or LPM megafunctions). In particular, the register design must include:
   a) An SR latch built from two cross-coupled NOR (not NAND) gates;
   b) A D latch built from this SR latch together with appropriate logic to drive the S and R inputs given D (data) and W (write) inputs;
   c) A rising-edge-triggered D flip-flop built from two of these D latches and appropriate logic to drive their W inputs given a clock input;
   d) An eight-bit register built from eight of the above D flip-flops in parallel.

You may, however, use the LPM_ADD_SUB megafunction to implement the 8-bit adder/subtracter that will be needed in your design.

5. Pre-Lab Preparation:

Reminder: You MUST complete the pre-lab assignment BEFORE coming to lab, and turn in your pre-lab report printout at the start of lab.

In this and future lab assignments, please continue to follow a structured design process, similar to the one you went through in detail in earlier lab assignments:

1. Identify the design context.
2. Specify the system-level requirements.
3. Create the system-level design.
4. Specify component requirements.
5. Create a testing plan.
6. Create the detailed design of the components.
7. Perform simulation testing of components.
8. Construct a prototype (in lab).
9. Test the prototype (in lab).

5.1. Guiding Questions

The following questions are intended to help guide and stimulate your thought process as you work through this assignment. As you gain experience, eventually you will be able to go through this type of thought process on your own, without being prompted.

Question #1. Think about how the user will provide the eight-bit input. What would be appropriate input devices to use for this purpose when prototyping the design on the DE2 board?

Question #2. The problem says that the input value should be added to (or subtracted from) a value stored in an accumulator register when an 'enter' button is pushed. Identify an input device that can serve as the button in question. Is this device active-high or active-low? And will your circuit take action when the button is *pressed*, or when it is *released*? (Either is OK, but make sure you specify which one you are trying to do, and build your design to match that specification.)

Question #2. Think about how the user will select whether the input value will be added to or subtracted from the stored value when the 'enter' button is pushed. Identify an appropriate device to use for this purpose.

Question #3. Think about how the output will be displayed. The problem says that both the input value as well as the current accumulator value must be displayed as unsigned hexadecimal numbers. Therefore, how many digits are needed to display each value? What specific output devices on the DE2 board would be suitable to use?

Question #4. Think about design-for-testability considerations. Until you know that your hex digit display is working, it might be a good idea to also display the values in a raw (e.g. binary) form. What devices on the DE2 board would be suitable to use for this purpose?

Question #5. Sketch the overall architecture of your system design as a block diagram. This top-level design must use (as sub-modules) the hex display and custom 8-bit register from the "extra requirements" (1) and (2) from section 4 above. (You may implement your top-level design as either a Quartus schematic or a VHDL file (your choice), as long as it uses these particular two entities which must be defined in VHDL and schematics respectively.) **NOTE:** You should make sure that your top-level design is such that, as required, the "current value" display *only* changes when the 'enter' button is pressed (or – also OK – only when it's released), and not at any other time, such as when the input value is being changed!

Question #6. For the VHDL hex byte display module, first think about how to decompose the function into two instances of a simpler entity, then think about how to code that lower-level entity. Here, aim to minimize your design effort, rather than to minimize the number of logic gates. For instance, you know that any arbitrary function of few input bits can be easily implemented as a lookup table. Think about how that lookup table would be implemented using standard modular combinational components. (You might even try implementing it that way initially, as a test.) What would be a corresponding language construct in VHDL? Write the design files, and test them in Quartus...

Question #7. To build the custom 8-bit register, start at the bottom with the custom SR latch, and work your way up. First, for an SR latch built from two cross-coupled NOR gates (as required), are the S (set) and R (reset) inputs active-high, or active-low? Work through some cases by hand or in the simulator to check. **NOTE:** Since this circuit contains feedback loops, a pure functional simulation of it will probably not work; instead you will have to do a full simulation, with timing delays included.

Question #8. Given the nature of the S and R inputs to your SR latch, what logic is needed to build the required D latch functionality on top of your SR latch? Recall that the inputs to your D latch should be *D* (the data value) and another bit *W*, which is an active-high signal that, when high, causes the latch output to be written (copied) from *D; i.e.,* the latch is "unlatched" or transparent. When *W* is low, the latch output should be quiescent (static); *i.e.*, the latch is latched. To avoid conflicts with Quartus' built-in Dlatch library symbol, call your module something else like "myDlatch".

Question #9. Next, build the 1-bit D flip-flop ("myDFF") using two copies of your D latch. The flip-flop should be rising-edge triggered, which means that the output should change to match the data input the instant after the clock first goes high, and at all other times, the output should be static (independent of the data input). Think about how the *W* control of each latch should be derived from the CLK input in order to achieve this behavior.

Question #10. Finally, build your custom 8-bit register using several of your custom DFFs in parallel. If you like, you can do this in several stages (*e.g.*, build the 8-bit register out of four 2-bit registers, or two 4-bit registers, *etc.*).

5.2. Pre-Lab Report Format

For this assignment, you may follow the generic pre-lab report format that is given in the *General Lab Requirements* document. This format is a little bit simpler than the one you had to use for lab #1. (Some of the larger projects you will do later in the semester will return to a more detailed format.) However, do be sure to include the simulation results, and the experiment design for the prototype testing.

6. In-Lab Procedure:

In lab this week, all you need to do is to put together and test your design, according to the experimental testing procedure that you already came up with in your pre-lab report. As you go along (at home and in lab), take detailed notes in your lab notebook (in accordance with the Lab Report Requirements section of the *General Lab Requirements* document) that you will base your final report on. Be sure to take note of anything that happens during prototype testing that is different from what you expected, and take detailed notes that you will use for analysis and interpretation of your experimental results in your final lab report. If any design modifications turn out to be needed, go ahead and make them, and re-test the design until it is working. Before leaving the lab, demonstrate to the TA that your circuit indeed does do the correct thing in a variety of input cases, and that it includes the required VHDL digit display module and the custom register design, and get the TA to sign off on your lab notebook. Remember to turn in your final lab report the following week.