EEL 3705L, Digital Logic Design Lab, Spring 2011

Lab Assignment #3: Design a 2-Digit Binary-to-Decimal Display Driver

Version history:
v0.1, 1/25/11: Copied Lab 2 v1.0 from Spr. ’07 to form this new assignment. –M. Frank
v1.0, 1/25/11: Merged in material from Lab 3 v1.0 from Spr. ’07. –MF

1. Document description:

This document specifies the third full lab assignment for the Spring 2011 semester of EEL 3705L (Digital Logic Design Lab) at the FAMU-FSU College of Engineering.

2. Assignment Synopsis:

In this lab assignment, students will utilize basic knowledge of Boolean algebra, binary numbers, radix number conversion, and logic minimization to design and optimize a modular combinational logic circuit which will convert input numbers in the range 0-99 from raw binary form into pairs of decimal digits that are visible to the user. This circuit can then be used as a component in later assignments. This lab is done as a structured design process, though with somewhat less documentation required than in lab #1. Students design and test their circuits at home in simulation for the pre-lab, and then test their designs in lab on the actual development boards.

3. Educational Objectives:

This assignment is intended to:

1. Exercise students’ understanding of base conversion.
2. Exercise students’ understanding of complex logic circuits including many gates.
3. Exercise students’ ability to minimize AND-OR circuits using K-maps.
4. Exercise students’ understanding of how to do modular design using existing components.
5. Give students practice at wiring up, simulating, and debugging such circuits using the Quartus tool.
6. 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 *modular* combinational logic circuit which will take as input a binary representation of a number from 0 to 99, and display the corresponding decimal digits in a form that is visible to and readily understandable by any (literate, but otherwise untrained) human viewer.

5. Pre-Lab Preparation:

Note: As usual, you MUST complete the pre-lab assignment BEFORE coming to lab, and turn in your pre-lab report at the start of lab.

First, here is a general outline of a structured design process, similar to the one you went through in detail in the previous lab assignment:

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

The above steps will serve as your model for this assignment.

5.1. Guiding Questions

The following questions are intended to help guide your thought process as you go 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 the design context that is implied by the design objective stated in section 4 above. The particular system that you will create (call it a "display driver") is part of some larger supersystem that includes some entities that provide inputs to your system, and entities that will process outputs from your system. In this case, the component that provides the input is unspecified – but you can still draw a block for it labeled "input device" in a block diagram for the supersystem. Similarly, the output device is unspecified, but you can draw a block labeled "output device." Go ahead and draw a system-level block diagram that shows these two blocks together with a block for your own component, and that shows the signals between them, and the names of the signals. The diagram will look pretty generic at first, but we will fill in more details about it as we go along.

Question #2. First, given that the input signal represents a number which can be any number from 0 to 99, what is the minimum number of bits that you will need to use in order to encode such an input? (Hint: What is 99 in binary?) Once you know the number of bits required, you can label the input bus in the diagram you drew in the previous question with its width in bits.

Question #3. Given the size of the input signal, what existing component(s) on the DE3 board could be used to conveniently provide this input in your design prototype? Decide precisely which devices on the board will be used to provide which input bits.

Question #4. What existing component on the DE3 board could be used, in your prototype, to conveniently display the output in a form that is visible to and understandable by the user? (Hint: What did you do in Lab #0 and Lab #1?)

Question #5. How many bits, at minimum, need to be sent to that component in order to display the desired digits? Knowing this allows you to label the output bus of the diagram you drew for question #1 with the required width.

Question #6. What will the various output digits look like, exactly, on the display device? Draw the appearances of all the digits, in an appropriate visual framework.

Question #7. At this point, you are in a position to specify the detailed requirements for your controller component. Go ahead and write them all down. The requirements should include a complete truth table. There should be 1 row in the truth table for each possible combination of input bits, and 1 output column for each output bit. Include “don’t cares” in the table as appropriate. This will go in the “requirements” section of your report. (NOTE: This is a very big truth table, but don’t worry! For later assignments, we’ll no longer do complete truth tables for our top-level designs.)

Question #8. Now that you have a truth table, although it has too many variables (how many?) to usefully draw K-maps for it, in principle you *could* still go ahead and use a more general systematic tabular method (like in §4.9 of the textbook) to manually create optimized AND-OR (SOP or POS) circuits for all the individual output bits. However, **DO NOT DO THIS!** We have now reached the point where monolithic hand-minimization of functions is no longer practical – it’s just too time-consuming! Instead…

Question #9. …we will use a modular approach. The key is *divide-and-conquer*. You could imagine breaking up the overall task into two simpler stages, such as: (1) Determine, from the input number, the numeric values of each of the two decimal digits making up the number, and (2) For each individual digit, translate its numeric value into the configuration of a particular unit of the overall hardware you are using to display both digits (this stage can be made of two identical copies of a single sub-component, since the mapping from digit value to display configuration should be the same for each digit). Sketch a block diagram for a modular system-level design that accomplishes this sort of breakdown of the overall task into stages.

Question #10. You’ll need to specify in detail the format & meaning of the signals that connect the module accomplishing stage (1) to the module(s) accomplishing stage (2). You’ll probably want to break the overall set of signals between the stages into groups representing each digit. What representation will you use for digits? How many bits does it require? Given this information, you can label the connections in the block diagram of the previous question with the appropriate bus widths. Include a section in your pre-lab report specifying exactly the meaning of all the bits in each connection.

Question #11. Now, what function needs to be performed to accomplish stage (1) – just breaking the number into two decimal digits? (Don’t worry at all yet, at this stage, about how each digit will be displayed – that is entirely the job of stage 2.) As a hint, think about the methods you learned for converting numbers between bases.

Knowing how to do base-conversion, can you express the value of each digit in terms of a simple arithmetical function of the value of the entire number? By a simple arithmetical function, I mean, for example, one or more of the following C-language operators: (+, -, *, /, %). (Make sure you know what *all* of these symbols mean.)

Question #12. From question 11, you should now have a simple expression for the value of each digit in terms of the value of the input number. At this point, you *could* design hardware to compute that function yourself. But that would be very time-consuming and error-prone. (Remember, we still have too many input bits to just draw a simple K-map!) Instead, you'd be wise to try to find an existing symbol-library component or megafunction that you can use directly to carry out the function that is needed. Look through what's available in the hierarchical list of libraries and LPM (Library of Parameterized Modules) megafunctions, using the Symbol tool in Quartus. You can find detailed documentation for megafunctions in Quartus's Help menu or through the main documentation page for Quartus (http://www.altera.com/literature/lit-qts.jsp) under Related Documentation → Using Megafunctions. Find a suitable existing component or components, state which one(s) you are using, and explain how you will tailor it/them to implement your stage 1 component. Include a functional specification of such component(s) in your pre-lab report (but you do not need to write out a full truth table for it/them).

Question #13. Now, for stage (2) from question #10, if breaking it down into two identical sub-components, each of which handles just one digit, what function does that sub-component need to implement? This particular function is small enough (how many inputs/outputs?) that you can write out its complete truth table and do a K-map for each output bit. This information can go into a "theory" subsection of the section of your pre-lab report that gives the detailed design of these sub-components.

Question #14. What is the detailed design of each component (except for any existing Altera components you may be re-using)? Draw each one as a schematic in Quartus. Put each module into a separate file. Create symbols for sub-modules and use them in higher-level modules. You can copy and paste your circuits into the pre-lab report.

Question #15. Think about how will you do simulation testing of the design? (I will go over the procedure for this in class on Monday.) Think about how you can simulate individual components by themselves, as well as the complete display driver system.

Question #16. How will you prototype the design? Identify what pin assignments that will be needed for prototyping your design. What will be your procedure for prototype testing?

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. Note that 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 display driver prototype, according to the experimental testing procedure that you already came up with in the pre-lab. As you go along, 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. If any design modifications turn out to be needed, go ahead and make them, and re-test until your design is working. Before leaving the lab, demonstrate to the TA that your circuit indeed does do the correct thing in all input cases, and get the TA to sign off on your lab notebook.

END OF DOCUMENT