EEL 3705/3705L, Digital Logic, Spring 2011

Lab Assignment #1: Structured Design Project #1: “Meeting Room Controller”

Version history:
v1.0, 1/10/07: Created initial version. –M. Frank
v2.0, 1/12/07: Inserted new sections 4 and 5 to the pre-lab report specs. –MF
Spr. ’11, v0.1, 1/13/11: Carried over v2.0 from Spr. ’07 to form starting point. –MF
v1.0, 1/13/11: Upgraded to slightly more complex “meeting room controller.” –MF

Document description:

This document describes the first full lab assignment for Spr. ’11. (There was a previous very simple warm-up “Lab #0” to familiarize the students with Quartus the boards.)

Assignment Synopsis:

In this lab assignment, students will take their first pass through a complete structured development process, although for a very simple project involving only a small number of logic gates. Students are given a simple description of the customer’s needs, and then at home they systematically go through the design process of translating those needs into detailed requirements, a system design, and a detailed design in an organized way. They test their design in simulation at home, and then test their the designs in lab on the actual boards. In later projects, they will go through this process again for larger projects.

Educational Objectives:

1. Exercise students’ understanding of basic logic gates such as AND, OR, XOR, and NOT.
2. Give students practice figuring out about how to manipulate both active-low and active-high signals in a single design.
3. Teach students how to place logic gates in Quartus and give them practice creating simple circuits using logic gates.
4. Teach students how to create test vectors and do functional simulation in Quartus.
5. Give students an opportunity to practice a structured design process for a very simple design.

Design Objective:

Here is the problem description (brief statement of the customer’s needs). This is much simpler than most real-world design problems would be, but it’s your very first logic design problem.

> **Design a logic circuit to control power to room lights and ceiling projector in a room with a motion sensor to provide an automatic power-off feature.**

Call this a "meeting room controller." The room has a video projector mounted on the ceiling, pointing to a large screen area on the front wall, which is covered with whiteboards, across the room from the door. There are 3 toggle switches on the wall: One switch next to the door, and two switches in a panel on the back wall next to the screen. The left switch on the back panel is labeled "front lights," and the right switch on that panel is labeled "projector." All 3 switches send continuous active-high signals to the controller while they are in the "up" position. There is a motion sensor in the corner of the room, which sends a continuous active-low "room empty" signal to the controller whenever no motion has been detected in the room any time in the last 10 minutes. The room has two sets of ceiling lights: A front set of 8 bright spotlights (along the front wall near the screen/whiteboards and lectern), and a back set of 18 dimmer lights across the middle of the room, and along the back wall, near the door. There are 27 power feeds (to each of the 26 lights and to the projector) which are controlled by individual logic signals from the controller; the signals controlling the lights are active-high, and the one to the projector is active-low. The controller's behavior should obey these rules:

1. Whenever the motion sensor reports the room is empty, power to all lights as well as to the projector should be turned off, to save energy, no matter what the state of the wall switches is.
2. Otherwise (whenever the room is not empty),
    a. The main (middle & back) room lights should be on if and only if the switch by the door is in the up position.
    b. The projector should have power if and only if the "projector" switch on the back panel near the lectern is in the up position.
    c. The bright front room lights should be on if and only if the switch by the door is in the up position, and the "front lights" switch by the lectern in not in the down position.

Pre-Lab Preparation:

Note: This week, you MUST complete the pre-lab assignment BEFORE coming to lab, and turn in your pre-lab report at the start of lab. Remember, pre-lab reports count as 25% of your lab grade!

First, recall from lecture 1 the general outline of a structured engineering design development process:

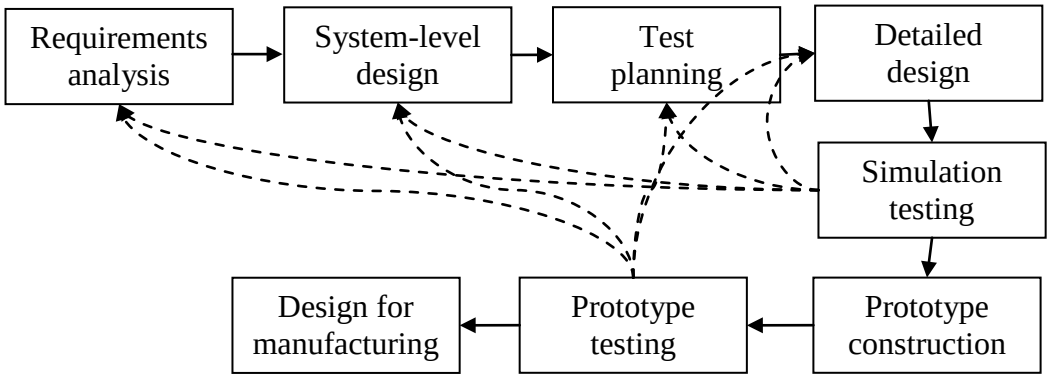

Figure 1. Structure of a generic engineering design development process. Dotted arrows indicate that if bugs are found during simulation or prototype testing, one may have to go back and redo some of the earlier steps in the design development process.

Before actually designing anything, the very first step an engineer must do is requirements analysis, where you translate the brief or informal problem statement from the customer into a detailed, organized list of specific requirements. Here are some questions to answer in order to help you think through the requirements analysis process.

Part 1. Design

Question 1. Our job is to design a control circuit for the power to various appliances (lights & projector) in a meeting room. First, how does this controller component fit into the larger system (the "supersystem") of which it is a part? Draw a schematic (block diagram) that includes a block (labeled rectangle) for each of the major components of the system, with a line for each connection between blocks.

Question 2. What are the names of the connections or interfaces between the major system components? Annotate your diagram from the previous question with the names of the signals. You should come up with appropriate, short but descriptive names.

Question 3. For each of those signals, in what direction does the information flow? Add arrowheads to your lines showing the direction of information flow.

Question 4. Define the electrical characteristics of each signal. What voltage ranges are to be used? (When no other information is given, for purposes of this class, you can assume that digital signals are standard TTL signals ranging between 0V and 5V. In the real world, you must check and make sure!) For each input, what is the input impedance? (Assume a high value for a digital input.) For each output, what is the required output current? (Assume a low value for a digital output.)

Question 5. What is the logical encoding of each interface signal? That is, for each signal, is active-high logic or active-low logic being used, and what exactly is the meaning of each of its possible signal values? Give a hierarchical list or table showing, for each interface signal (input or output), its name, whether it is an input or an output, which component it comes from, and goes to, its voltage range, its possible values, and a description of their meanings.

Question 6. What are the formal functional requirements on your controller, given the information in the problem statement and the signal definitions in question 5? Write your requirements as a detailed hierarchical list, and as a table showing the input/output combinations for the controller. Give the I/O ports (pins or pin buses) of your controller meaningful-sounding names.

Now, here are some questions to walk you through the system-level design process.

Question 7. Inside your controller system, what will its major top-level components be? You don't have to decide yet what specific logic gate(s) you are using in each one. Instead, just think about what are the different functions your controller will need to perform. Divide up the whole job of doing these functions into pieces that can be carried out by simpler subsystems, and name each one. Give a top-level block diagram for your controller, like you did for the entire system in questions 1-3. Include all the connections between the controller's internal components and the input/output ports that connect it to the rest of the supersystem.

Question 8. Now, for each of the top-level components inside your controller, put requirements on it. These just follow from the controller's structure, and the requirements on the whole controller from question 6.

Question 9. Now, for each of the top-level components inside your controller, what is its detailed internal design? For this problem, this may be very simple, a few logic gates, or perhaps even just a single gate. Sketch the design in your lab notebook and in Quartus. The example lab report from a previous year's lab (smoke alarm controller) may give you some ideas.

Part 2. Implementation

Now that you've figured out how your system components should be implemented, we'll walk you through the steps to actually create and wire up logic gates in Quartus.

Step 1. Create a new folder and a new project and schematic file for your controller circuit, like you did in Lab0. Call it Lab1. Go ahead and put down the input and output ports you will need. You don't have to assign them to pins yet – we will do that later.

Step 2. It's up to you what logic gates to put down. Click the button that looks like an "AND" gate from the toolbar at the left, or just double-click on the grid. This will bring up the "Symbol" dialog. In the "Libraries" area, expand "primitives" and then "logic" and you will see a list of available gates. If you click on the gate name, you will see a picture of it to the right in the dialog. Some gates that you might want to look at and consider using in your design are `not`, `and2`, `band2`, `bnand2`, `bnor2`, `bor2`, `nand2`, `nor2`, `or2`, `xnor`, and `xor`. There are a variety of possible gates, and combinations of gates, that could be used to meet your specifications. If going through this list gives you any new ideas for your design, you can go back and redo your answer to question 9.

Anyway, after selecting the gate you want to try, click "OK" and place it in your design. Repeat this process for each gate you want to put down.

Step 3. Wire up the gates appropriately to the inputs and outputs according to your design.

Step 4. This step does not affect the function of the circuit, but is important for documentation purposes. Select the rectangle tool from the schematic drawing toolbar, and draw a rectangle around the gate(s) implementing each individual sub-component of your controller. Then select the text tool (the button looks like a letter "A"), click near the rectangle of each component, and type the name of the component from your original system-level design in question 7. This makes it clear from your schematic what are the major top-level components of your design. In later labs, you will learn how to put each component into its own file.

Step 5. Now your design is complete. Save your schematic and compile it. Don't worry about pin assignments yet (they are not needed for compilation).

Part 3. Simulation Testing

In this part of the assignment, we walk you through the process for testing your design under the simulator, to make sure it will produce the outputs you expect.

Step 6. Under the "Processing" menu, select "Simulator Tool." For simplicity, if you do not care about the speed of your circuit, you may want to select "Simulation mode: Functional" and hit the "Generate Functional Simulation Netlist" button. Click the "Open" button near the bottom of the resulting dialog to start creating your input waveforms (also known as "test vectors"). In the pane on the left side of the "<filename>.vwf" window that comes up, right-click and select "Insert Node or Bus…". Click "Node Finder…". Click "List" to see all your circuit nodes. Select the input nodes (use shift-click for multiple-select), click the ">" button to add them to the list of selected nodes on the right, and then click "OK." Hit OK again in the "Insert Node or Bus" window. Now your input nodes should appear on the left side in the waveform window.

Step 7. Now we will actually create the waveforms for the input nodes. Select the first input node. If it's an active-low input, then in the toolbar on the left, select the "Forcing High" button (labeled with a 1) to model the input normally being high (1). The waveform will shift up a little. Do the same for your other inputs. Now, select the Waveform Editing Tool (third tool from the top), and drag out some time interval for the first input, say 10-30 ns. This will toggle the state of that input during that time interval. Do the same for some (perhaps partially overlapping) time interval for the second input, say 20-40 ns. And so forth. For a small design like this, you should make sure that the resulting waveform includes all possible combinations of 1's and 0's for all inputs. You can do this more easily by selecting all the inputs, grouping them together (right-click and select Grouping → Group… from the popup menu, then enter a name for the group), then select the group and click the "Count Value" button from the left toolbar (has a "C" in it), and hit OK – this will automatically generate all possible values for you.

Step 8. Close the waveform window, and save changes as Lab1.vwf. Click "Start Simulation" in the Simulator Tool window. After the simulation finishes, you should get a pop-up dialog saying "Simulator was successful." Now click the "Report" button. A "Simulation Waveforms" window should come up.

Step 9. Look at the waveforms corresponding to your output pins, and compare them with the input waveforms. If your design was correct, then the output waveforms should do what the problem specified – although with some delay, unless you selected functional simulation.

Question 10. Looking at the waveforms, about how much delay is there from the controller inputs to the corresponding outputs? (It should be 0 if you did a functional simulation only.)

It would be a good idea at this point to go back and revise your requirements to specify a minimum acceptable delay for the controller's outputs – although of course, the actual delay (which is only a few nanoseconds) will probably be very small in comparison.

Part 4. Prototype Testing

For this part, think about how you will test your design on the actual DE2 prototyping board. Notice that you could use three slider switches to provide a mock-up of the inputs from the wall switches, and a pushbutton (say) to model the signal from the motion detector, and individual LEDs in the LED row to represent the room lights (Hint: How many red LEDs are there on the board? How many green ones?). You could use one of the 2-symbol 7-segment display panels to display the word "On" when the projector is turned on. The following questions will help guide you through this process.

Question 11. You already know how the switches and 7-segment display work from lab #0. As for the pushbuttons and the individual LEDs, do they use active-low or active-high signals? Look up the answers in the DE2 board user guide.

Question 12. Given your answer to question 11, will you use additional circuitry to convert between the pushbuttons/LEDs and the inputs to your controller, or will you simply access these devices directly?

Question 13. What are the formal names and pin numbers of the board-level signals you will use to access each of your I/O devices? Look these up in the user guide and pin assignment file, and make a table of them.

Question 14. When you come into lab, what procedure will you use to systematically test all of your design's functional requirements? Describe each step of this process.

At this point, you should be more or less ready for the lab. Before showing up at the lab, write up your design and testing plan in the pre-lab report, **which must be turned in at the start of lab**! The format for the pre-lab report is described below.

Pre-Lab Report Format

This time, your pre-lab report will have a slightly different format than the one given the general lab requirements. The idea here is to organize what you did in answering the questions above, and put it in the form of a professional design document. Please divide your pre-lab report into the following sections:

1. Introduction

Give a brief, general introduction to the design problem.

2. Design Context

1.1. Supersystem design

In this section, give the top-level system block diagram for the entire room control system that you prepared in questions #1-3. In your text, point out which of the top-level subsystems you are in charge of designing.

1.2. Interfaces between supersystem components

In this section, describe the interfacing signals between the top-level components of the room control system, as addressed in questions #2-5 above.

1.3. System-Level functional behavior requirements

In this section, give a list of requirements as to how the overall system is supposed to behave. *I.e.*, take the descriptions of functions from the original problem description, and put them in a list. Number each item ("Requirement #1," "Requirement #2," etc.) In addition to the list, also useful in this case would be a table, showing, for each possible combination of circumstances, what the system's overall behavior ought to be.

3. System Requirements Specification

In this section, use the information in the previous section and the original problem description to define the requirements for the particular subsystem you're designing.

3.1. Interface Requirements

In this section, describe your subsystem's interface to the outside world.

3.1.1. General Interface Requirements

Here, list the requirements that apply to all of the input/output signals – this will save you having to list them separately for each signal later. In this case, the general requirements include things like the voltage levels, active-low or active-high logic, etc. Also state whether positive or negative logic is being used. (That is, is a logic 1 considered a high voltage or a low voltage?)

3.1.2. Input Signal Requirements

In this section, state the specific requirements for all input signals to your component. This includes all the relevant information from questions 2-5.

3.1.3. Output Signal Requirements

Same thing, but for the outputs.

3.2. Functional Behavioral Requirements

In this section, you list the detailed requirements regarding all of the specific functions or behaviors that your specific component is supposed to carry out. These follow from the behavioral requirements on the overall system. You can give these in a list. Also useful

in this case would be a table, showing, for each possible combination of input signal value, what output signal values should be produced.

4. Top-Level Design for Controller Subsystem

In this section, give the top-level design of your controller subsystem. This includes your answers to questions 7 and 8 above. Include the following subsections:

4.1. System Architecture

This includes your block diagram for the controller; how is it broken down into components? What are the I/O signals for each component?

4.2. Component Functional Requirements

Give the detailed functional requirements for each component. These follow from the requirements on the entire controller component from section 3, and on how you broke it down into components.

5. Detailed Design of Controller Components

This includes your answers to question 9. Copy and paste your detailed schematics from Quartus. Explain your choice of gates, and explain why the circuits should work.

6. Testing Plan

In this section, you plan and design computer simulation experiments and prototype testing experiments that you will carry out in order to test your subsystem design, once it is built.

4.1. Simulation testing plan

In this section, describe, in your own words, the step-by-step process for testing your controller system electronically under Quartus. (Please don't just copy my detailed instructions for using the simulator from above, but just summarize the process in your own words.)

4.2. Prototype construction plan

In this section, describe in detail how you will put together your prototype in lab. This includes the procedure for programming the Cyclone II chip on the DE2 board, and wiring up the pins. Make sure you describe your pin assignments.

4.3. Prototype testing plan

In this section, describe in detail the testing procedure you developed in question #14.

7. Answers to Questions

Here, give your answers to the questions #1-14 from above. Much of this material is redundant with the earlier sections, but you can just copy and paste as needed.

8. Conclusion

Write this as stated in the *General Lab Requirements.*

In-Lab Procedure:

In lab this week, all you need to do is to put together and test your controller prototype according to the experimental testing procedure that you already came up with in the pre-lab. As you go along, take detailed notes of what you did and what the results were in

your lab notebook (in accordance with the Lab Report Requirements section of the *General Lab Requirements* document) that will enable you to write up your final report. Be sure to take note of anything that happens that is different from what you expected, and take notes 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 the correct thing in all input cases, and get them to sign off on your lab notebook. Your final lab report will be due at the start of lab the following week.