ECE241F - Digital Systems
University of Toronto

Lab #3 - Fall 2008
Logic Optimization, 7-Segment Displays and Incremental Design

1. Introduction

The purpose of this laboratory is to gain experience in manual logic optimization. The exercise involves the design of a circuit that converts a code represented by input switches into characters on a display, like those on a digital watch. This lab will also illustrate the key engineering concept of *incremental* design, in which small portions of a circuit are first created and tested, and then used to build larger circuits.

2. Preparation

The preparation for this lab is to read Parts I, II and III in section 3 and implement the designs as indicated, using the Quartus II software you learned about in Lab #2. Your preparation (which must be pasted or taped into a lab book, and ready at the beginning of the lab) should consist of the following:

1. For Part I below you are asked to design a circuit that drives a 7-segment character display. This circuit requires *seven* different logic functions, one for each segment in the display. For the preparation you are to present to the teaching assistant the Karnaugh map that you used to optimize each of these seven functions.
2. In Part I below you are to use Verilog code to describe the 7-segment display circuit. This code is the second part of the preparation.
3. For each of Parts I through III you will be asked to design a number of circuits. For each of these circuits you must create a suitable simulation (using Quartus's simulator) of your design. Your preparation must include a printout of the simulation results for each of these circuits. You must write comments on your simulation waveforms that explain what is being tested at each point in time. You must also indicate the thought process you used in testing these circuits - i.e. what test cases you chose to simulate to prove that the circuit works, and why.

3. Lab

Parts I to III below require that you implement a number of Quartus II projects. To make this a little easier, a starter kit has been provided on the course website, found at:

`http://www.eecg.utoronto.ca/˜jayar/ece241_08F/Lab3_starterkit.zip`

The starter kit is a ZIP archive containing a Quartus II project for each part of the lab. Unzip the archive into a working directory of your choice.

The in-lab component of your lab grade will be based on the circuit you create for Part III below; this is the only part of the exercise that you need to show to the teaching assistant. Note, however, that Parts II and III of the lab reuse the results from Part I, so you will need to work through all of Parts I through III. A bonus mark (9/8 is the maximum grade) can also be obtained if you complete Part IV. Please inform your TA when you have completed your work and are ready to be graded.

Part I - Design of a 7-Segment Decoder Circuit

Figure 1 shows a 7-segment character display controlled by a logic circuit. These displays are common on digital watches and various electrical devices. This circuit is called a 7-segment *decoder* and it has a three-bit input $c_2c_1c_0$ (which you will connect to switches on the DE2 board), and 7 outputs that turn on or off the 7 different segments (lights) in the character display. The three inputs present a code that are translated by the circuit into 7 outputs to create a particular character by lighting up some of the lights on the display.

You are to design a circuit that is able to decode and display *the last 5 digits of your student number*. An example mapping is given in Table 1, which lists the characters that should be displayed for each value of $c_2c_1c_0$. *However, make sure to replace the sequence* 12345 *with the last 5 digits from your student number*. To keep the design simple, you only need to decode five numbers, represented using the codes $c_2c_1c_0$ =000, 001, 010, 011, and 100. The fifth code should produce a blank character with all of the lights off. Since we don't care what is displayed for the remaining code values (110 and 111) you can be treat these as don't care values (recall that don't care values are values which will not occur, and thus can be leveraged to simplify your logic expression).

The seven segments in the display are identified by the numbers 0 to 6 shown in Figure 1. Each segment is lit up by driving it to the logic value 0 (which is a little opposite of what you might expect). You must implement a separate logic function that controls each segment in the display. Use a Karnaugh map to determine the minimal sum-of-products expressions for each of these 7 outputs.

Create a Verilog module for your 7-segment decoder circuit. In your Verilog code, use only simple **assign** statements to specify each of the sum-of-products expressions generated from your Karnaugh maps. In your Verilog code assign the $c_2c_1c_0$ inputs to switches SW_{2-0} on the DE2 board, and assign the outputs of the decoder to the HEX0 display on the DE2 board (as you learned in Lab #2). The segments in this display are called HEX0_0, $\text{HEX0}_1, \ldots, \text{HEX0}_6$, corresponding to Figure 1.

Recall from Lab #2 that you need to to include pin assignments in your Quartus II projects so that the switches and lights used in your logic circuit are assigned to the appropriate pins on the FPGA chip which are physically connected to the switches and character displays. The pin assignments for the HEX0 display are listed in the pin assignment file that we introduced in Lab 2, which is called *DE2_pin_assignments.csv*. In this pin assignments file the HEX0 segments are declared as an array. To use the same names in your Verilog code, your should declare the seven-bit output port as

output [0:6] HEX0;

By using this 7-bit array, the names of the seven segments in your code will match the names that are used for these segments in the *DE2_pin_assignments.csv* file.

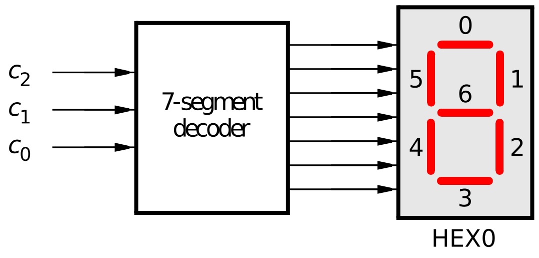

Figure 1. A 7-segment decoder.

$c_2c_1c_0$	Character
000	1 (replace with last digit of your student number)
001	2 (replace with 2nd last digit of your student number)
010	3 (replace with 3rd last digit of your student number)
011	4 (replace with 4th last digit of your student number)
100	5 (replace with 5th last digit of your student number)
101	'blank'
110	'Don't Care'
111	'Don't Care'

Table 1. Character codes (for the case where your student number ends in '12345')

During the lab, compile and download your circuit onto the DE2 board. Test the functionality of the circuit by toggling the SW_{2-0} switches and observing the 7-segment display HEX0.

Part II - Selecting the Character to be Displayed

Consider the circuit shown in Figure 2. It uses a three-bit wide 5-to-1 multiplexer (just like the one you built in Lab #2) to enable the selection of five characters that are displayed on a 7-segment display. Using the 7-segment decoder from Part I this circuit can display any of the five digits and 'blank'. The character codes are set according to Table 1 by using the switches SW_{14-0}, and a specific character is selected for display by setting the switches SW_{17-15}.

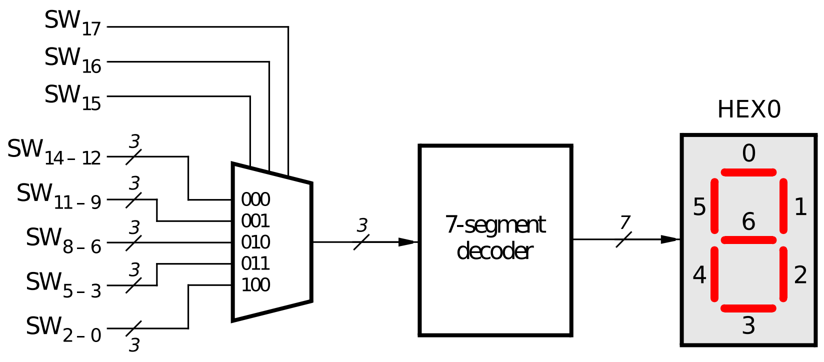

Figure 2. A circuit that can select and display one of five characters.

To build this circuit you can reuse two smaller designs already created: the three-bit 5-to-1 multiplexer from Part IV of Lab 2 and the 7-segment decoder from Part I of this lab. Your task is to create a new *top-level* design that implements the circuit in Figure 2 using the multiplexer and decoder as subcircuits. You are to design this top-level circuit twice, using two different approaches:

1. Create the top-level design using the Quartus II schematic capture tool. In this case you use Quartus II to create a schematic symbol for the multiplexer and decoder circuits, and then include these symbols in the top-level schematic, as described in Tutorial Section B.5.1 on page 847 of the text. Note that the multiplexer and decoder are still implemented by using your Verilog code, and it is only the top-level design that is described by drawing the schematic.

2. Create the top-level design by writing complete Verilog code for the circuit in Figure 2. An outline of this code, which shows how to include the multiplexer and decoder subcircuits in the top-level Verilog module, is given in Figure 3.

To design the top-level module by using schematic capture perform the following steps.

1. The project for this part is provided in the starter kit. Open the Quartus II software and then use the File > Open Project command to open the project named *hierarchy_schematic.qpf* in the *part2/hierarchy_schematic* subdirectory to begin your work.
2. In the part2 folder you will find the Verilog files *mux3bit_5to1.v* and *char_7seg.v*. Edit these files and put in the code you previously wrote to implement the three-bit 5-to-1 multiplexer and the 7-segment decoder, respectively.
3. The next step is to use Quartus II to create a *symbol file* for each of the above Verilog files. The symbol file allows the corresponding Verilog module to be used as a subcircuit in your top-level schematic. The procedure for generating a symbol file and including it in a schematic is shown in section B.5 of the course textbook.
4. In Quartus II use the command File > New to create a new Block Diagram file and draw the circuit in Figure 2. Insert the symbols for the subcircuits created above, insert input and output ports, and draw the wiring connections shown in Figure 2.

   In your schematic diagram you will need to create vector ports to attach to the switches and 7-segment displays. You do this in Quartus II by specifying the name of a signal as a vector of the form NAME[X..Y]. For example, in the schematic capture tool SW_{2-0} is designated using the input pin name SW[2..0] and $\mathrm{HEX0}_{0-6}$ is denoted using an output port named HEX0[0..6].
5. Include the required pin assignments (you can use the *DE2_pin_assignments.csv file*) for the DE2 board switches and 7-segment display. Compile the project.
6. Simulate the compiled circuit using a timing simulation in the Quartus II Simulator.

To complete this lab using Verilog code perform the following steps.

1. The project for this part is provided in the starter kit. In Quartus II open the project named *hierarchy_verilog* in the *part2/hierarchy_verilog* subdirectory to begin your work.
2. In the part2 folder you will find the verilog files *mux3bit_5to1.v* and *char_7seg.v*. Edit these files and put in the code you previously wrote to implement the three-bit 5-to-1 multiplexer and the 7-segment decoder, respectively.
3. Use the File > New command to create a new Verilog file named *hierarchy_verilog.v*. Type your top-level Verilog code, following the style of code shown in Figure 3, into this file.
4. Include the required pin assignments for the DE2 board switches and 7-segment display. Compile the project.
5. Simulate the compiled circuit using a timing simulation in the Quartus II Simulator.

Both simulations (of the schematic-based and Verilog-based projects) can be done using the same input waveforms and should produce identical outputs.

```
module hierarchy_verilog (SW, HEX0);
    input [17:0] SW;         // toggle switches
    output [0:6] HEX0;       // 7-seg displays

    wire [2:0] M;

    mux_3bit_5to1 M0 (SW[17:15], SW[14:12], SW[11:9], SW[8:6], SW[5:3], SW[2:0], M);
    char_7seg H0 (M, HEX0);
endmodule

// implements a 3-bit wide 5-to-1 multiplexer
module mux_3bit_5to1 (S, U, V, W, X, Y, M);
    input [2:0] S, U, V, W, X, Y;
    output [2:0] M;

    . . . code not shown

endmodule

// implements a 7-segment decoder for each character
module char_7seg (C, Display);
    input [2:0] C;             // input code
    output [0:6] Display;      // output 7-seg code

    . . . code not shown

endmodule
```

Figure 3. Verilog code for the circuit in Figure 2.

Download your circuit (you can choose either the schematic version or the Verilog version for this testing phase) onto the DE2 board. Test the functionality of the circuit by setting the proper character codes on the switches SW_{14-0} and then toggling SW_{17-15} to observe the display of characters.

Part III - Displaying and Rotating a Sequence

In this part you will reuse your previous designs, and extend the code from Part II so that it uses **five** 7-segment displays, rather than just one. You will need to use five instances (copies) of the subcircuits from Part II. The purpose of your circuit is to display a sequence on the five displays, and be able to manually rotate this sequence in a circular fashion across the displays when the switches SW_{17-15} are toggled. As an example, if the displayed sequence is 12345, then your circuit should produce the output patterns illustrated in Table 2. The sequence must be the last five digits of your student number.

SW_{17} SW_{16} SW_{15}	HEX4	HEX3	HEX2	HEX1	HEX0
000	1	2	3	4	5
001	2	3	4	5	1
010	3	4	5	1	2
011	4	5	1	2	3
100	5	1	2	3	4

Table 2. Manually rotating the sequence 12345 on five displays.

Perform the following steps.

1. The project for this part is provided in the starter kit. Open the project named *part3* in the *part3* subdirectory to begin your work.

2. Copy the Verilog files *mux3bit_5to1.v* and *char_7seg.v* from Part II into the project directory for Part III. Use the Quartus II command Project > Add/Remove Files in Project to add these files to the *part3* project.

3. Create a new Verilog file called *part3.v* and write the code to instantiate the required subcircuits (no modifications to the subcircuits are necessary). Connect the switches SW_{17-15}, in the same order, to the select inputs of each of the five instances of the three-bit wide 5-to-1 multiplexers. Also connect SW_{14-0} to each instance of the multiplexers as required to produce the patterns of characters shown in Table 2. Each multiplexer will share the same set of inputs (SW_{14-0}), but the inputs will connect to each multiplexer in a different manner. It is up to you to arrange them properly such that you can manually "rotate" your numbers. Following this, you must connect the outputs of the five multipexers to the 7-segment displays HEX4, HEX3, HEX2, HEX1, and HEX0 on the DE2 board.

4. Include the required pin assignments for the DE2 board switches and 7-segment displays. Compile the project.

5. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by setting the proper digit codes on the switches SW_{14-0} and then toggling SW_{17-15} to observe the rotation of the digits.

Part IV (optional/bonus of 1)

Extend your design from Part III so that is uses all eight 7-segment displays on the DE2 board. Your circuit should be able to display the last five digits of your student number on the eight displays, and manually rotate the displayed sequence when the switches SW_{17-15} are toggled. If the displayed sequence is 12345, then your circuit should produce the patterns shown in Table 3.

SW_{17} SW_{16} SW_{15}	HEX7	HEX6	HEX5	HEX4	HEX3	HEX2	HEX1	HEX0
000				1	2	3	4	5
001			1	2	3	4	5	
010		1	2	3	4	5		
011	1	2	3	4	5			
100	2	3	4	5				1
101	3	4	5				1	2
110	4	5				1	2	3
111	5				1	2	3	4

Table 3. Rotating the sequence 12345 on eight displays.

Perform the following steps:

1. The project for this part is provided in the starter kit. Open the project named *part4* in the *part4* subdirectory to begin your work.

2. Write the required Verilog code for this part of the exercise and include these Verilog modules in the Quartus II project. Connect the switches SW_{17-15} to the select inputs of each instance of the multiplexers in your circuit. Also connect SW_{14-0} to each instance of the multiplexers as required to produce the patterns of characters shown in Table 3. (Hint: for some inputs of the multiplexers you will want to select the 'blank' character.) Connect the outputs of your multiplexers to the 7-segment displays HEX7, . . ., HEX0.

3. Include the required pin assignments for the DE2 board switches and 7-segment displays. Compile the project.

4. Download the compiled circuit into the FPGA chip. Test the functionality of the circuit by setting the proper character codes on the switches SW_{14-0} and then toggling SW_{17-15} to observe the rotation of the characters.