CPE 2211 COMPUTER ENGINEERING LAB

EXPERIMENT 4 LAB MANUAL

Revised: J. Tichenor FS19

SEVEN SEGMENT DECODER DESIGN

OBJECTIVES

In this experiment you will

- Gain experience with use of Karnaugh Maps.
- Gain practical experience with seven segment displays.
- Use the suite of EDA tools supplied by Altera: Quartus-II and ModelSim.

LAB REPORTS

The format of lab reports should be such that the information can be used to reproduce the lab, including what values were used in a circuit, why the values were used, how the values were determined, and any results and observations made. This lab manual will be used as a guide for what calculations need to be made, what values need to be recorded, and various other questions. The lab report does not need to repeat everything from the manual verbatim, but it does need to include enough information for a 3rd party to be able to use the report to obtain the same observations and answers. Throughout the lab manual, in the Preliminary (if there is one), and in the Procedure, there are areas designated by **Qxx followed by a question or statement**. These areas will be **bold**, and the lab TA will be looking for an answer or image for each. These answers or images are to be included in the lab report. The lab TA will let you know if the lab report will be paper form, or if you will be able to submit electronically.

PURPOSE

Develop experience with the use of Karnaugh maps and gain some familiarity with seven segment displays. Another goal of this exercise is to introduce you to designing with Altera field programmable gate arrays. You will design a simple seven segment decoder for displaying four bit hexadecimal numbers on a standard seven segment LED display. You will verify the design by using schematic capture to enter the design for logic simulation. After your design is functioning correctly, you'll download it to the Cyclone II FPGA. It is assumed that you are already familiar with schematic capture and logic simulation so you will not be guided through that process like before.

REFERENCES

Tutorial covered in Lab 2: Introduction to Digital Design with Using Schematic Capture with Altera QuartusII and Simulation using ModelSim-Altera.

BACKGROUND

Seven segment displays are a common item in electronic systems. If you are wearing a watch with a digital display you probably have several with you. Light emitting diodes (LEDs) and liquid crystal displays are two common technologies which are used to implement these components. The configuration shown in Figure 1 is a standard way that the seven segments are labeled as segments a through g.

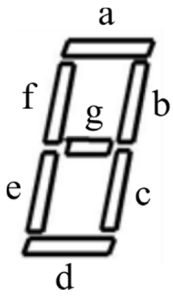

Figure 1: Seven segment display.

Seven segment displays constructed from LEDs almost always have either the anode or cathode of each LED wired together. This is referred to as a common anode or common cathode seven segment LED. Common cathode displays are called ***active high*** since the common cathode is tied to ground, each segment anode tied to a segment driver (gate) through a current limiting resister, and a logic high output turns the segment on. Assuming a positive logic convention this means that logic "1" turns the segment on. Common anode displays are called ***active low*** since the common anode is tied to the positive power supply, each segment cathode tied to a segment driver through a current limiting resister, and a logic low output turns the segment on. Assuming a positive logic convention this means that logic "0" turns the segment on. Seven segment displays intended for use with TTL devices are generally active low since TTL can sink more current than it can source. Seven segment displays intended for use with CMOS devices can be either common anode or common cathode.

PRELIMINARY

This portion should be completed before you come into the lab. Do not attempt to complete all of this during the lab period.

1. Develop a four input seven output truth table for the segment drivers of an ***active low*** seven segment display which will display all of the hexadecimal digits. Your display should use the font shown in Figure 2 below for the sixteen hex digits.

Figure 2: Segment representation of the sixteen hex digits.

The truth table in Figure 3 shows two aspects of the design. First, by comparing the □ representation in the DISPLAY column to the segment designation of Figure 1, it can be seen that the 'g' segment is the only segment turned off (***active low***, so a 0 turns on a segment, and 1 turns it off). Secondly, the 'a' column of Figure 3 can be determined by

comparing each DISPLAY element to the segment designation of Figure 1 for segment ‘a’.

Q1. Finish filling out the truth table.

DISPLAY	SWITCHES/INPUT				SEVEN SEGMENTS OF LED FROM a TO g						
	w	x	y	z	a	b	c	d	e	f	g
0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1						
2	0	0	1	0	0						
3	0	0	1	1	0						
4	0	1	0	0	1						
5	0	1	0	1	0						
6	0	1	1	0	0						
7	0	1	1	1	0						
8	1	0	0	0	0						
9	1	0	0	1	0						
A	1	0	1	0	0						
b	1	0	1	1	1						
C	1	1	0	0	0						
d	1	1	0	1	1						
E	1	1	1	0	0						
F	1	1	1	1	0						

Figure 3: Truth table to complete.

2. Make Karnaugh maps for the truth table in filled out in Figure 3. The Karnaugh Map for segment ‘a’ is shown in Figure 4, where it can be seen that the ‘a’ column from Figure 3 has been translated to the Karnaugh Map.

WX\YZ	[00]Y'Z'	[01]Y'Z	[11]YZ	[10]YZ'
[00]W'X'	0	1	0	0
[01]W'X	1	0	0	0
[11]WX	0	1	0	0
[10]WX'	0	0	1	0
	a = w'x'y'z+w'xy'z'+wxy'z+wx'yz			

Figure 4: Karnaugh Map for segment ‘a’.

Q2. Write out the Karnaugh Maps for the remaining segments b, c, d, e, f, and g.

Q3. Write minimum two level SOP expressions for each segment driver using the Karnaugh Maps developed.

Q4. Write minimum two level POS expressions for each segment driver using the Karnaugh Maps developed.

You should use care at this stage of your design. Errors are easy to correct here and get harder and more time consuming to correct as you progress. If you are working as a small group you might consider splitting up the work and checking each other’s work. Check your design carefully. You might use techniques learned in class by developing either a truth table or logic equation from your circuit and proving that your circuit realizes your original truth table or logic equation. You should end up with a fairly neat, hand drawn schematic on paper that has a fighting chance at working correctly after you enter it using design architect.

PROCEDURE

Your goal is to implement the SOP logic obtained for each segment using parts from the primitive's gate in library. The main difference between your paper design and the actual hardware design is a) you will need to provide input ports and output ports as shown in Figure 5 to connect your design to the outside world and b) you'll need to wire up your FPGA design to switches and lights so you can provide inputs and view results. Step b) will be completed in Lab 5.

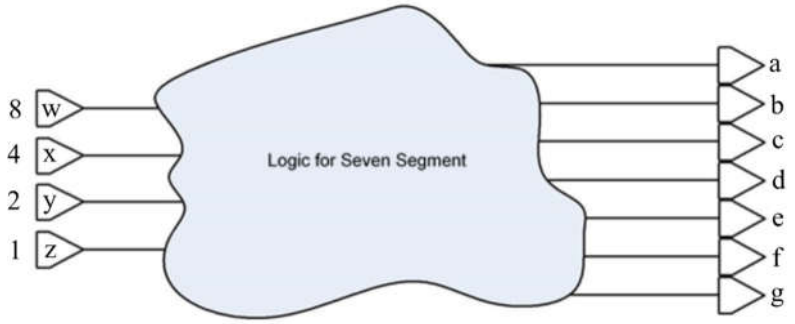

Figure 5: Schematic.

1. Open Altera Quartus-II, and create a new project called "Lab4". Design a block diagram to create binary to seven segment decoder. The name of this entity should be *"Lab4."* The schematic for this decoder is based on the SOP you wrote in the Preliminary.
2. Name the input-output ports, save and compile the top level entity.
3. Create VHDL code from Schematic.
4. Open ModelSim and create a new project named "Lab4", add Lab4.vhd in the existing file.
5. Recall that in lab 2 the force command was used to define the inputs to the schematic. However, here there are 4 input bits which will result in 16 input combinations and defining these test patterns using the force commands would require considerable amount of time. Instead, the clock command will be used to define the test inputs. Assume that each input combination will last for 100ps, so the total length of the simulation will be 1600ps. By looking at the completed truth table of Figure 3, that was assigned for the preliminary it can be seen that the least significant bit of the input changes every 100ps and hence 16 times. The next higher order bit changes 8 times or every 200ps, the bit higher in order to that changes 4 times or every 400ps and finally the MSB changes only twice or every 800ps.

   In the wave window, right click on the input '1' or 'z' depending on how it was named, and select ***clock,*** this will bring a window box as shown in Figure 6. After setting the ***Period*** to be 100 and ***First Edge*** to be Falling, click on OK. Repeat this procedure for the remaining inputs.
6. Type 'run 1600 ps' in the command window, you should be able to see the following waveform in Figure 7.

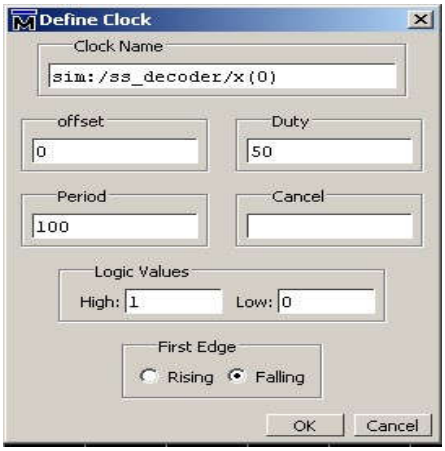

Figure 6: Clock definition for input '1', or 'z'.

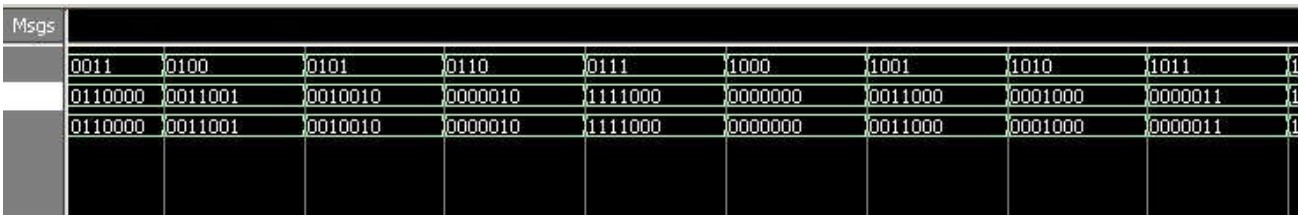

Figure 7: Simulation output.

Q5. Record screen capture of schematic.

Q6. Record screen capture of Wave workspace.

Q7. Demonstrate on the Wave image that the input translates to the correct output.

Q8. What is the total number of primitive gates you used?

Q9. Determine the worst case propagation path from any input to any output of your design.