

EXPERIMENT NUMBER 4 SEVEN SEGMENT DECODER DESIGN

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: Introduction to Digital Design with Using Schematic Capture with Altera Quartus-II and Simulation using ModelSim-Altera.

Materials Required

Altera Quartus-II, ModelSim-Altera

Preliminary

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

Background

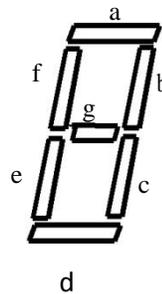


Figure 4-1: Seven segment display

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 4-1 is a standard way that the seven segments are labeled as segments a through g.

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 4-1 is a standard way that the seven segments are labeled as segments a through g.

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 resistor, 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 (5v), each segment cathode tied to a segment driver through a current limiting resistor, 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.

Design

- a) 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 4-2 below for the sixteen hex digits.

0 1 2 3 4 5 6 7 8 9 A b c d e f

Figure 4-2

- b) Make Karnaugh maps for the truth table in part a.
 c) Write minimum two level SOP expressions for each segment driver using the Karnaugh maps in part b.
 d) Write minimum two level POS expressions for each segment driver using the Karnaugh maps in part b.
 e) Implement the segment drivers using primitive gates.

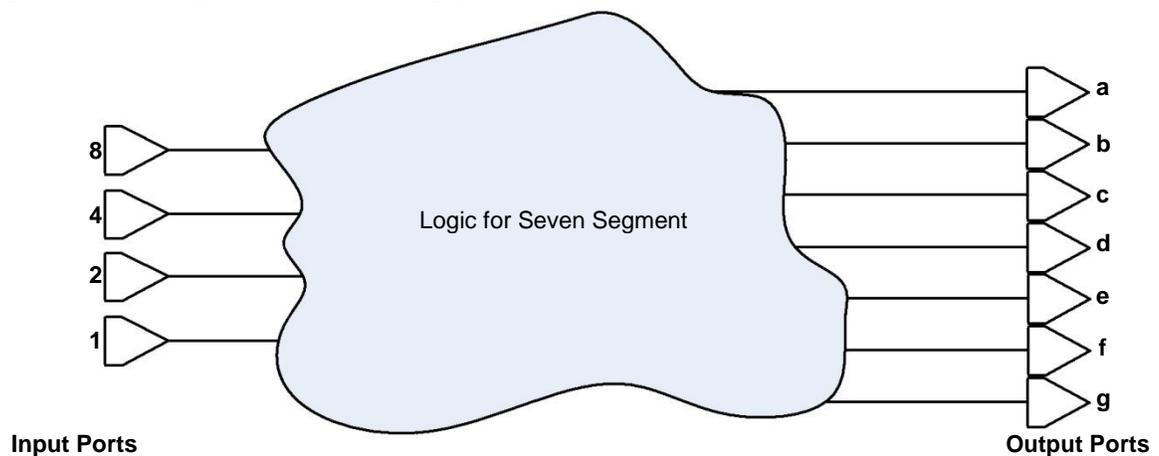


Figure 4-3: Schematic for Lab 7

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 primitives 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 4-3 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.

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. Simulation within setting the different clock to the input. Before we get to observe the output we have to define our inputs "8", "4", "2" and "1". Recall that in lab 2 we used the force command to define the inputs to our schematic. However, here we have 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. We will use the clock to define our test inputs.

We assume that each input combination will last for 100ps, so the total length of our simulation will be 1600ps. If we look at the truth table of the SSD that was assigned for the prelims we will realize 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" and select *clock*, this will bring a window box as shown in figure 4-4. After setting the period to be 100 and First edge to be Falling, click on OK. Repeat this procedure for the inputs "2", "4" and "8".

6. Type "run 1600 ps" in the command window, you should be able to see the following waveform in Figure 4-5.

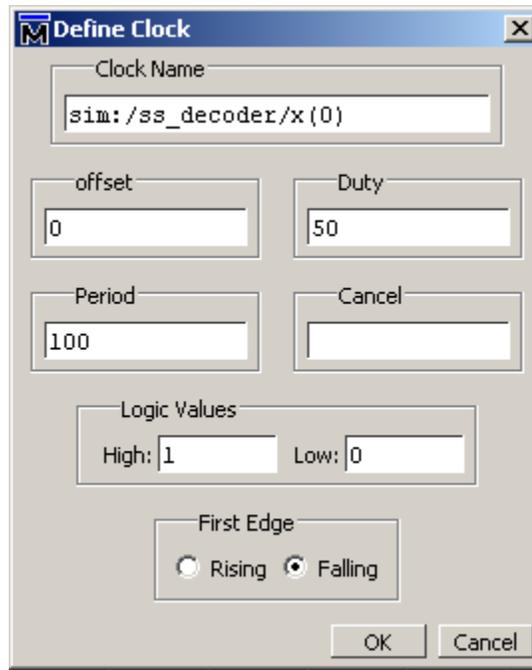


Figure 4-4.

Msgs									
	0011	0100	0101	0110	0111	1000	1001	1010	1011
	0110000	0011001	0010010	0000010	1111000	0000000	0011000	0001000	0000011
	0110000	0011001	0010010	0000010	1111000	0000000	0011000	0001000	0000011

Figure 4-5.

Questions

1. What is the total number of primitive gates you used?
2. Determine the worst case propagation path from any input to any output of your design.