ENGR 272 Lab 3: Combinational Logic (Seven Segment Driver)

3.1 Background Information

Seven segment displays are used to display numbers in many electronic devices, such as alarm clocks, VCRs, microwaves, and many more. They are a popular choice because they are cheap, easy to read, and easy to program.

In simple seven-segment displays like the one used in this lab, each segment is controlled individually, and only one digit is displayed at any given time. In this section, only one digit will be used, so this limitation will not be a problem, but all four digits will be used in future sections, which will require some extra logic to cycle through the digits.

A digital logic decoder is a circuit that converts coded inputs into coded outputs. A seven-segment decoder is a simple example of a "4-to-7 decoder", taking a 4-bit binary number in and outputting the signals to control each of the 7 segments. Read through Example 2.10 from the course textbook (Page 79) for an example of a seven-segment decoder implementation.

In digital logic design, there are tools and methods used to make simplifying logic much easier. A Karnaugh Map is a way to turn any truth table into simplified logic. Textbook section 2.7 Example 2.10 shows the logic for an active high 7-Segment decoder. The decoder designed in section 3 of the lab is an active low decoder to match our hardware. In an active high system, the LED is lit when the logic level is high. In an active low system, the LED is lit when the logic level is low.

3.2 Section Overview

Objective: Design and implement a decoder on the FPGA to convert a 4-bit binary number inputted using the switches into a single digit of hexadecimal on the seven-segment display.

1. Create a block diagram for your design.
2. Fill out Figure 3.2, which shows how each number 0-F will be displayed on the seven-segment display.
3. Fill out Table 3.1, the functional truth table for the seven-segment decoder.
4. Create seven Karnaugh Maps (K-Maps), one for each segment, to minimize the logic for each segment using the sum-of-products canonical form.
5. Simulate the design of your seven-segment decoder.
6. Program and test hardware.

3.3 Learning Objectives

In this section, the following new items will be covered:

1. Using Karnaugh Maps to create minimized Boolean logic equations corresponding to a functional truth table.
2. Translating Boolean logic from equations to logic gates.

3.4 Materials

1. Quartus Prime Lite Edition V. 18.0
2. DE10-Lite kit with MAX10 10M50DAF484C7G FPGA
3. USB to USB-B cable

3.5 Procedure

There are 6 steps to digital logic design:

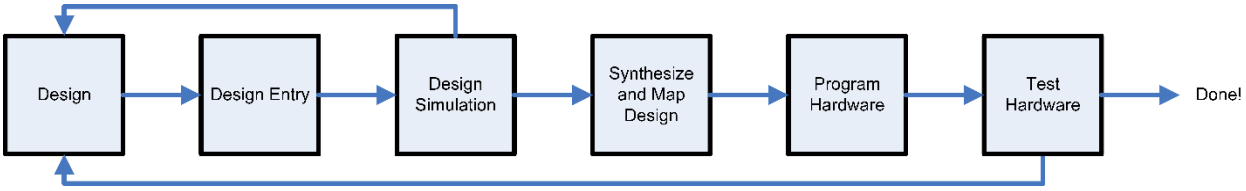

Figure 3.1: Use this process for designing the seven-segment display.

1. **Design:** The context of the design is established in this step. The context involves defining the inputs, desired outputs, and all the logic required in-between. In this step, all the minimizations and layout are planned for the design entry process. While this step is not always the longest, it should involve the most thought and effort. This typically requires a complete block diagram showing all the logic blocks and the connections between them, often with written explanations of specific functions.
2. **Design Entry:** The actual drafting of the digital logic design occurs in this step, translating the design from block diagrams and descriptions into the software. This can be accomplished directly by writing HDL code, or graphically by drawing a schematic that a software tool can convert into HDL code.
3. **Design Simulation:** Before committing to hardware, this step tests the design in a controlled computer simulation. If the design does not function as specified in the “Design” step, it is revised.
4. **Synthesize and Map Design:** When the design simulates correctly, the HDL and schematic source files are synthesized into a design file that can be written to the FPGA. This includes assigning the inputs and outputs of the design to IO pins.
5. **Program Hardware:** After the design file is created, it is used to configure the FPGA. Quartus Prime sends a bit stream over the USB-B cable to configure the DE10-Lite FPGA.

6. **Test Hardware:** Verify hardware operation once the FPGA has been programmed. The FPGA should operate exactly as the design predicted, which was verified by the simulation. Synthesis problems, timing violations, or incorrect assumptions about the hardware can require the designer to return to the “Design” step.

3.6 Design

Figure 3.2 shows all of the different outputs that the seven-segment decoder should produce. Shade in the segments that should be on for each input, 0-F. You can use figure 2.48 from the textbook to help complete all Hexadecimal characters 0-F. For the letters, A-F use upper case letters except for B and D.

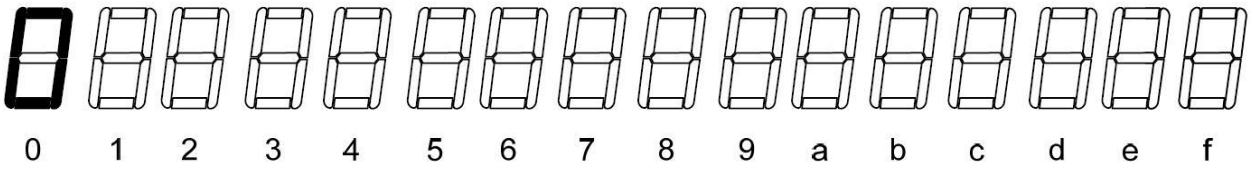

Figure 3.2: Displaying hexadecimal numbers on a seven-segment display

3.6.1 Make a block diagram

1. Make a block diagram showing how your design goes together.
2. Add all connections and Pin Names.

3.6.2 Make a functional truth table

Use the shading in Figure 3.2 and the diagram in Figure 3.3 to fill in the functional truth table below. You may find it useful to use Table 2.6 from the textbook to reflect the figures created using active low hardware.

The seven-segment display is active-LOW, meaning that sending a 0 to a segment turns it ON, and sending a 1 to a segment turns it OFF!

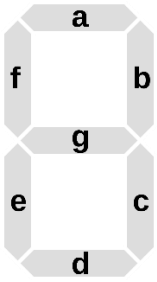

Figure 3.3: Seven-segment display segments labeled

Input (Hexadecimal)	Input (4-bit Binary)	SegA	SegB	SegC	SegD	SegE	SegF	SegG
0	0000							
1	0001							
2	0010							
3	0011							
4	0100							
5	0101							
6	0110							
7	0111							
8	1000							
9	1001							
a	1010							
b	1011							
c	1100							
d	1101							
e	1110							
f	1111							

Table 3.1: Seven-segment decoder functional truth table

Make sure to double check the above table for accuracy. Errors this early in the design can lead to a lengthy debugging process.

3.6.3 Minimize the logic

1. Create seven K-Maps, one for each segment.
2. Determine the sum-of-products minimized Boolean equations for each segment using the K-Maps. **Even though the seven-segment display is active-LOW, you would still group the 1's for the minimization.**

3.7 Design Entry

Enter the logic gates corresponding to each minimized Boolean logic equation determined in Section 3.6.3 into a schematic. Enter the design using the same process as in Section 1.6 and be sure to use good design practices to keep the schematic neat! Debugging a poor layout can be very difficult.

3.8 Design Simulation

Simulation is done with ModelSim in this course using the same process as in Section 1.7. See the links in Moodle for information on the software. Your simulation results should match Table 3.1.

To generate the Verilog Hardware Description Language file for use in Modelsim, go to File -> create/update -> create HDL Design File for current file -> Verilog (Just like lab 2, If you broke your design up into different modules, you will need to do this for each module).

Find and open the **.v** file/files in Modelsim and simulate your design.

3.9 Synthesize and Map Design

Synthesize your design and assign pins to your inputs and outputs using the same process as in Section 1.8.

3.10 Program Hardware

Program the DE10-Lite using the same process as in Section 1.9.

3.11 Test Hardware

Test to see if the switches cause the correct outputs on the seven-segment display.

3.12 Checkoff

1. Figure 3.2 filled out
2. Table 3.1 filled out
3. A K-Map and minimized Boolean logic equation for each segment
4. Complete schematic for the seven-segment decoder
5. Simulation matches Table 3.1
6. Hardware output matches simulation results

3.13 Study Questions

The study questions go at the end of the Lab Report

1. When is a simulation necessary? Was it useful for this section?

3.14 Challenge - Extra Credit

Display the result of the adder from Lab 2 onto the seven-segment display.

3.15 Report Rubric

The reports for the labs are to be formal reports and are to be completed using a word processing application. Please turn in the report to Moodle in a PDF or MS Word format (No Google Docs share links).

Your report should contain at a minimum the following items:

1. **Title**
2. **Date**
3. **Name**
4. **Objectives**
5. **Equipment/Parts/Materials**
6. **Procedure**
    a. **Design**
        i. **Figure 3.1 filled out.**
        ii. **Table 3.1 filled out.**
        iii. **Karnaugh Maps used to create the minimized Boolean logic equations.**
    b. **Design Entry**
        i. **Block Diagrams**
        ii. **All schematics you created in Quartus Prime (Screen Captures).**
        iii. **A screen capture of what pin assignments you made in the Assignment Editor in Quartus.**
    c. **Design Simulation**
        i. **ModelSim simulation figures (Screen Captures).**
7. **Complete the study questions in section 3.13**
8. **Challenge - Extra Credit (If this was done)**
9. **Observations**
10. **Conclusion**
11. **References**
12. **Appendix (Source code of Verilog files used for your simulations).**

All images need to have a title and a figure number. For more detailed information about what goes into each of the different sections, see “ENGR 272 Guidelines for Technical Lab Reports” in Moodle.

For an example of what your reports should look like see “ENGR 272 Example Report” in Moodle.

References:

1. Oregon State University ECE 272: Section 3: Combinational Logic (Seven Segment Driver) (2019) https://eecs.oregonstate.edu/tekbots/courses/ece272/section3

2. Harris, S. L. & Harris, D. H. (2016) Digital Design and Computer Architecture: ARM Edition 1st Edition, Waltham, MA: Elsevier.