ECE 272 Lab 3
Fall 2018

Combinational Logic (Seven-Segment Driver)
Phi Luu

October 24^{th}, 2018
Grading TA: Edgar Perez
Lab Partner: Benjamin Geyer

1 Introduction

Seven-segment displays are used in various electronic devices, such as microwave, digital clock, and video cassette players. They are one of the most popular choices to display numbers and most letters on digital devices due to their easy programming and user interfaces.

A seven-segment display has seven LEDs arranged in the shape of the digit 8. Each segment has its own signal name, and all segments connect to the decoder separately. There can be an additional LEDs which has a circle shape next to the digit, representing a decimal place. Figure 1 shows what a seven-segment display looks like.

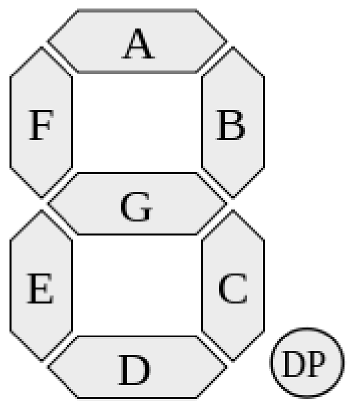

Figure 1: The LED segments and their order in a seven-segment display[1]

In this lab, I and Ben design a seven-segment display capable of displaying hexadecimal digits. We make a decoder that convert a 4-bit binary number into a single digit on the seven-segment display. The LED layout of each of the hexadecimal digits is as follows in Figure 2:

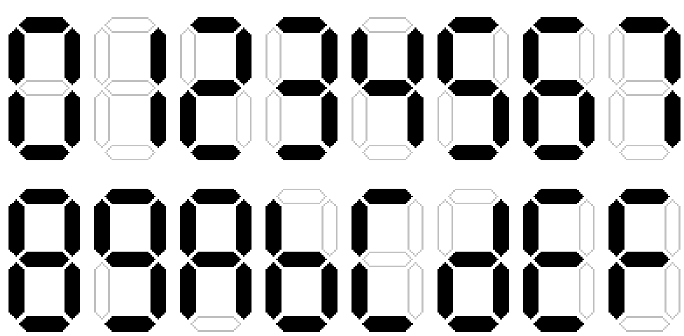

Figure 2: Seven-segment display showing hexadecimal digits [2]

2 Design

We use four switches to represent 4-bit binary numbers as inputs and a seven-segment display—or 7 LED segments—as outputs. To make outputs change accordingly (and correctly) to different values of the inputs, our objective is to use combinational logic to program a seven-segment display decoder on the FPGA. A visual representation of our objective is illustrated by the block diagram in Figure 3.

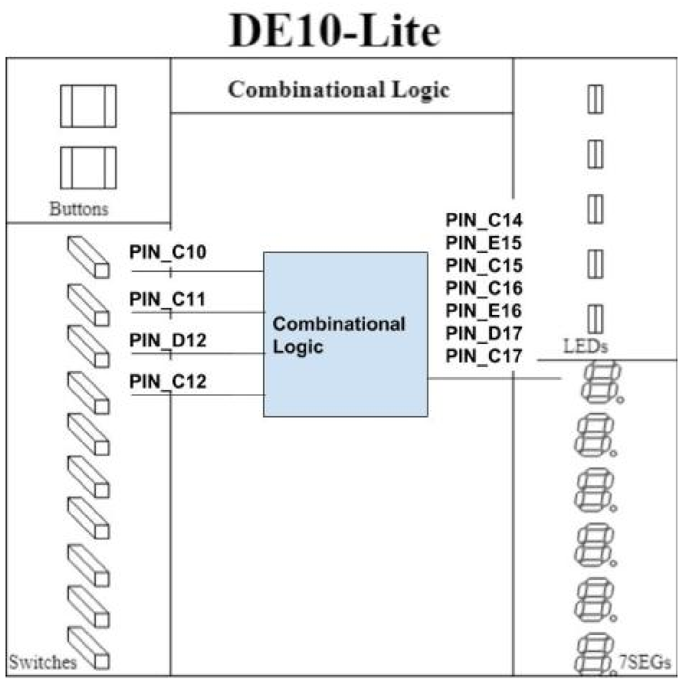

Figure 3: Block diagram

Since the LED segments on the display are active-low—which means the pins will be active if low voltage is applied to it, the LED segment will turn on if the output signal is 0 and turn off if the signal is 1. Using the segment layout and digit syntax in Figures 1 and 2, we construct the following truth table:

Input (Hexadecimal)	Input (4-bit Binary)	$\textbf{Seg}_{\textbf{A}}$	$\textbf{Seg}_{\textbf{B}}$	$\textbf{Seg}_{\textbf{C}}$	$\textbf{Seg}_{\textbf{D}}$	$\textbf{Seg}_{\textbf{E}}$	$\textbf{Seg}_{\textbf{F}}$	$\textbf{Seg}_{\textbf{G}}$
0	0000	0	0	0	0	0	0	1
1	0001	1	0	0	1	1	1	1
2	0010	0	0	1	0	0	1	0
3	0011	0	0	0	0	1	1	0
4	0100	1	0	0	1	1	0	0
5	0101	0	1	0	0	1	0	0
6	0110	0	1	0	0	0	0	0
7	0111	0	0	0	1	1	1	1
8	1000	0	0	0	0	0	0	0
9	1001	0	0	0	0	1	0	0
a	1010	0	0	0	1	0	0	0
b	1011	1	1	0	0	0	0	0
c	1100	0	1	1	0	0	0	1
d	1101	1	0	0	0	0	1	0
e	1110	0	1	1	0	0	0	0
f	1111	0	1	1	1	0	0	0

Table 1: Conversion table between hexadecimal, 4-bit binary, and seven-segment decoder

Using Table 1, the digit 0 in hexadecimal system is equivalent to 0000 in binary system. Segments A through F in the decoder are lit up, and segment G will be turned off—this is exactly what Figure 2 illustrates. The remaining rows of the table are interpreted in the same way.

From Table 1, we can write a sum-of-product form for each of the segments using their minterms. **Note that even though the segments are active-low, the Boolean equations still use rows with output 1 (since they are the minterms, and minterms always use 1).**

We want to simplify the Boolean equation of each of the segments as much as possible so that we can use minimal hardware when building the circuit of the decoder. Therefore, we use Karnaugh maps to simplify the Boolean equations of the segments, as demonstrated on the next page.

Seg_A

CD \ AB	00	01	11	10
00	0	1	0	0
01	1	0	1	0
11	0	0	0	1
10	0	0	0	0

$$Seg_A = \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}D + A\overline{B}CD$$

Seg_B

CD \ AB	00	01	11	10
00	0	0	1	0
01	0	1	0	0
11	0	0	1	1
10	0	1	1	0

$$Seg_B = BC\overline{D} + ACD + AB\overline{D} + \overline{A}B\overline{C}D$$

Seg_C

CD \ AB	00	01	11	10
00	0	0	1	0
01	0	0	0	0
11	0	0	1	0
10	1	0	1	0

$$Seg_C = ABC + AB\overline{D} + \overline{A}\,\overline{B}C\overline{D}$$

Seg_D

CD \ AB	00	01	11	10
00	0	1	0	0
01	1	0	0	0
11	0	1	1	0
10	0	0	0	1

$$Seg_D = BCD + \overline{A}\,\overline{B}\,\overline{C}D + \overline{A}B\overline{C}\,\overline{D} + A\overline{B}C\overline{D}$$

Seg_E

CD \ AB	00	01	11	10
00	0	1	0	0
01	1	1	0	1
11	1	1	0	0
10	0	0	0	0

$$Seg_E = \overline{A}D + \overline{A}B\overline{C} + \overline{B}\,\overline{C}D$$

Seg_F

CD \ AB	00	01	11	10
00	0	0	0	0
01	1	0	1	0
11	1	1	0	0
10	1	0	0	0

$$Seg_F = \overline{A}CD + \overline{A}\,\overline{B}D + \overline{A}\,\overline{B}C + AB\overline{C}D$$

Seg_G

CD \ AB	00	01	11	10
00	1	0	1	0
01	1	0	0	0
11	0	1	0	0
10	0	0	0	0

$$Seg_G = \overline{A}\,\overline{B}\,\overline{C} + \overline{A}BCD + AB\overline{C}\,\overline{D}$$

There are a few Boolean terms that appears in more than one Boolean equations, such as $\bar{A}B\bar{C}\bar{D}$ and $\bar{A}\bar{B}\bar{C}D$ in segments A and D. Hence, the logic gates can be reused, which further simplifies the hardware needed to build the circuit. A schematic of the circuit is as in Figure 4 below.

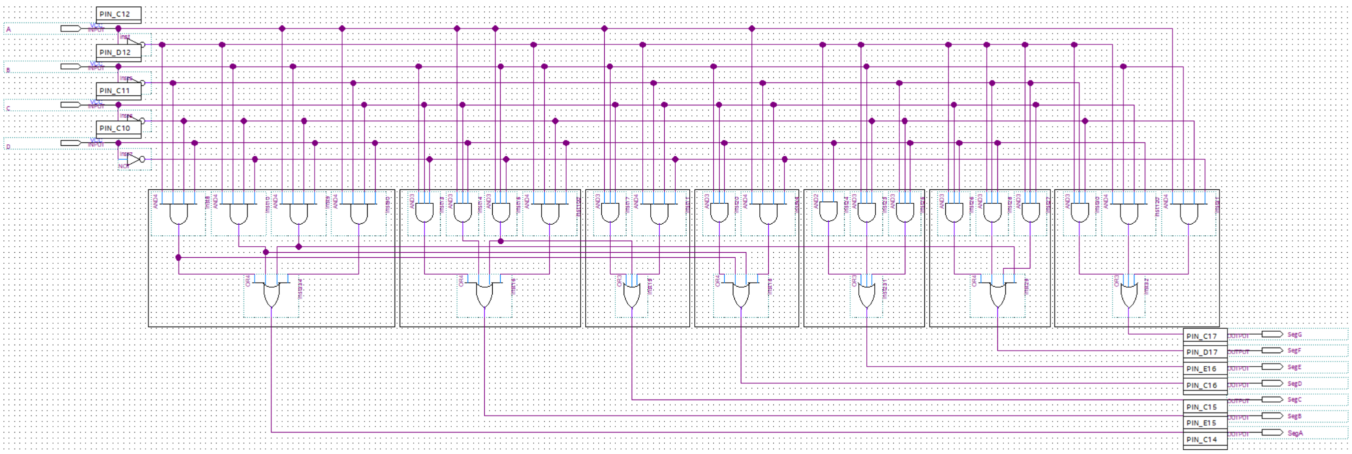

Figure 4: A schematic for the seven-segment display decoder. Due to the large difference between the size of the schematic and the available space, an image with higher resolution has been uploaded here.

The schematic of the decoder is indeed complicated, yet it can be easier read when divided into smaller groups. A simpler way to read the schematic is that there are seven group of combinational logic gates and 8 input buses (4 bits of input and their inversions).

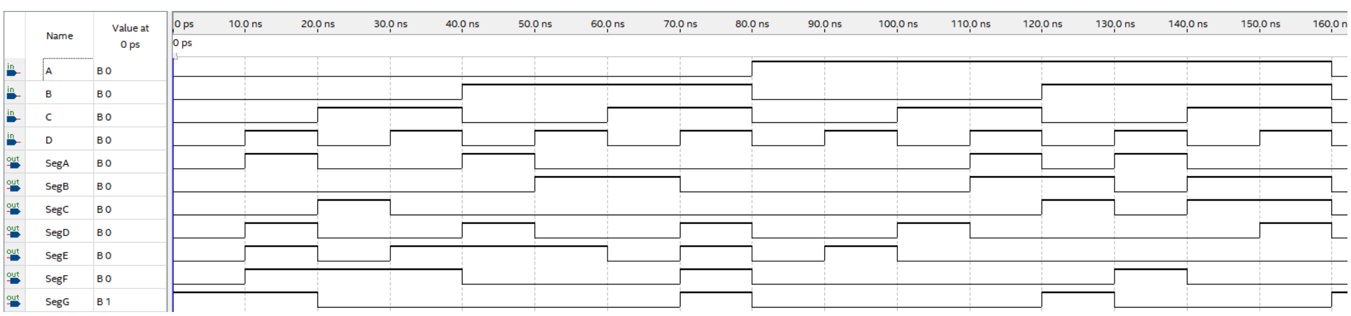

Figure 5: Simulation waveform of the program with each 10-nanosecond interval representing a hexadecimal digit

3 Results

4 Experiment Notes

Reflection

Study Questions

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

Appendix

No appendix is available in this lab.

References

[1] W. Commons, “Seven segment display.” https://commons.wikimedia.org/wiki/Seven_segment_display, 2018.

[2] E. E. S. Exchange, “Hex to 7 segment decoder for a common anode 7 seg display.” https://electronics.stackexchange.com/questions/373034/hex-to-7-segment-decoder-for-a-common-anode-7-seg-display, 2018.