ECE 272 Lab #3

Combinational Logic (Seven Segment Driver)

Marshall Saltz

11/04/2022

1. Introduction

The purpose of this lab is to design and implement a 4:7 decoder on the DE10-Lite FPGA to convert a 4-bit binary input to a single digit hexadecimal number on the FPGA's active low display. I will be using K-maps to create minimized Boolean logic equations for each output that corresponds to a truth table. The hardware used is a MAX10 FPGA, a USB to USB-B Cable, and my computer. The software used is Quartus Prime Lite version 18, ModelSim, and Microsoft Word to record the truth table values and work through the Karnaugh Maps.

The seven segment display that is my output is an active low device, meaning each segment is turned on when they receive a 0 instead of a 1. The decoder I am designing is a circuit that takes coded inputs and converts them to coded outputs. This decoder is a 4-to-7 decoder, meaning it takes a 4-bit binary number and outputs the signals to control seven segments individually.

2. Design

Before I started the lab, I drew out what I wanted my output to look like.

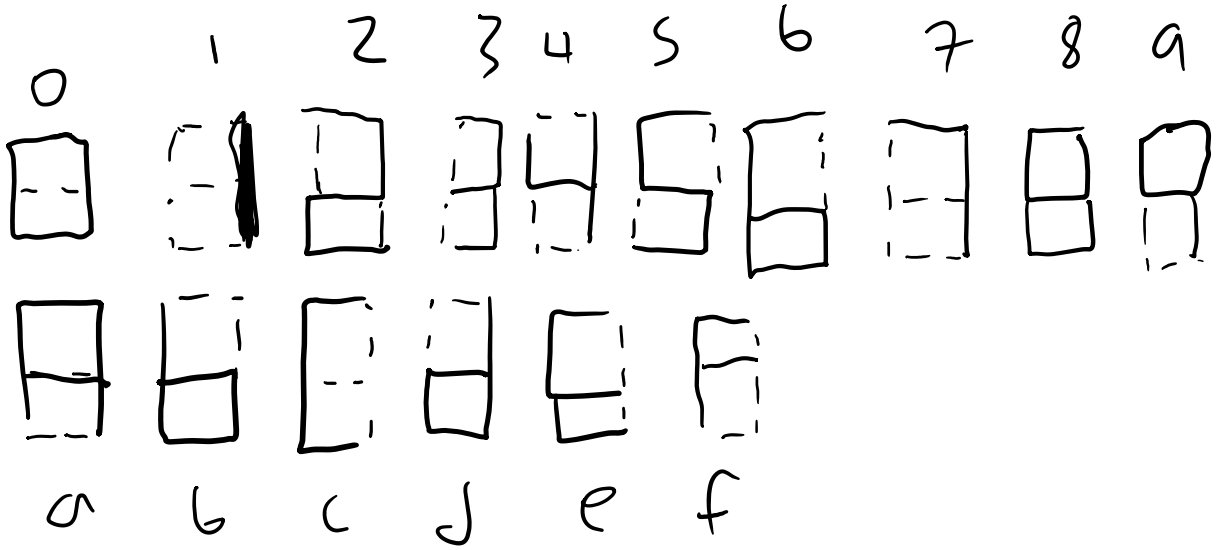

Fig. 1: Intended Output

This is the intended output, or what I would like to see when I run the program on the board.

Next, I drew out the truth table that would provide this output.

Input_16	Input_2	S_A	S_B	S_C	S_D	S_E	S_F	S_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	1	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

Fig. 2: Truth Table

This truth table takes in different binary numbers and converts them to hexadecimal numbers on the seven segment display. Each segment is one output, and they are all active low.

Next, I drew out the Karnaugh Maps for each segment.

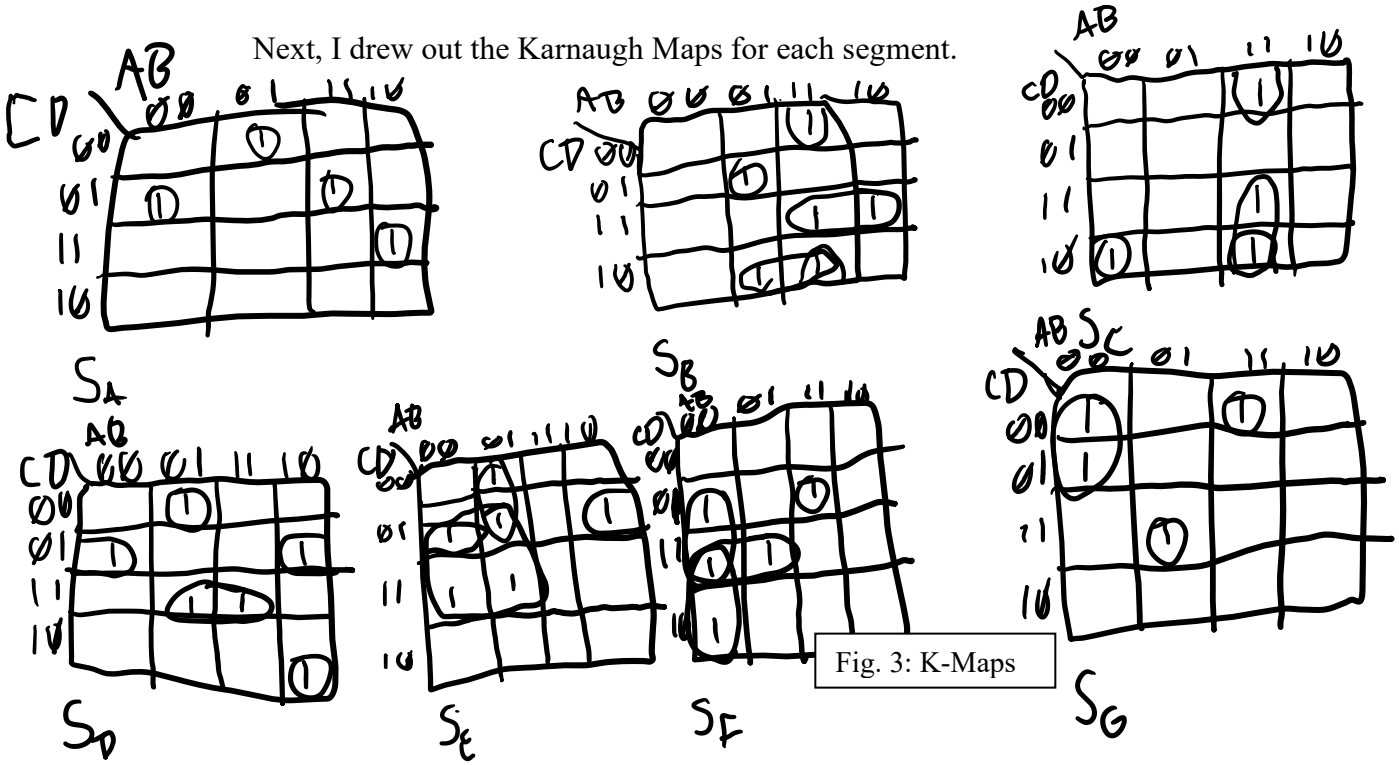

Fig. 3: K-Maps

Fig. 3 shows the Karnaugh Maps I drew from the truth table. These helped me derived minimized Boolean equations for each of the seven outputs.

$$S_A = A'BC'D' + A'B'C'D + ABC'D + AB'CD$$
$$S_B = A'BC'D + ABD' + ACD + BCD'$$
$$S_C = A'B'CD' + ABD' + ABC$$
$$S_D = A'BC'DD + AB'CD' + B'C'D + BCD$$
$$S_E = A'D + A'BC' + B'C'D$$
$$S_F = ABC'D + A'B'D + A'B'C + A'CD$$
$$S_G = A'B'C' + ABC'D' + A'BCD$$

Fig. 4: Boolean Equations

Above are the Boolean equations I wrote for each segment, derived from the Karnaugh maps in Fig. 3.

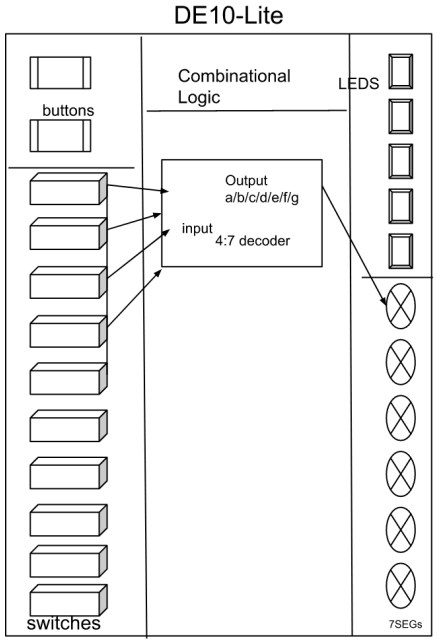

Fig. 5: Combinational Logic

Fig. 5 shows the combinational logic for the decoder. It takes in four inputs as switches and has one of the seven segment displays as an output.

Next, I proceeded to convey these Boolean equations in a schematic. Below are screenshots of individual pieces of the schematic as well as an overall image.

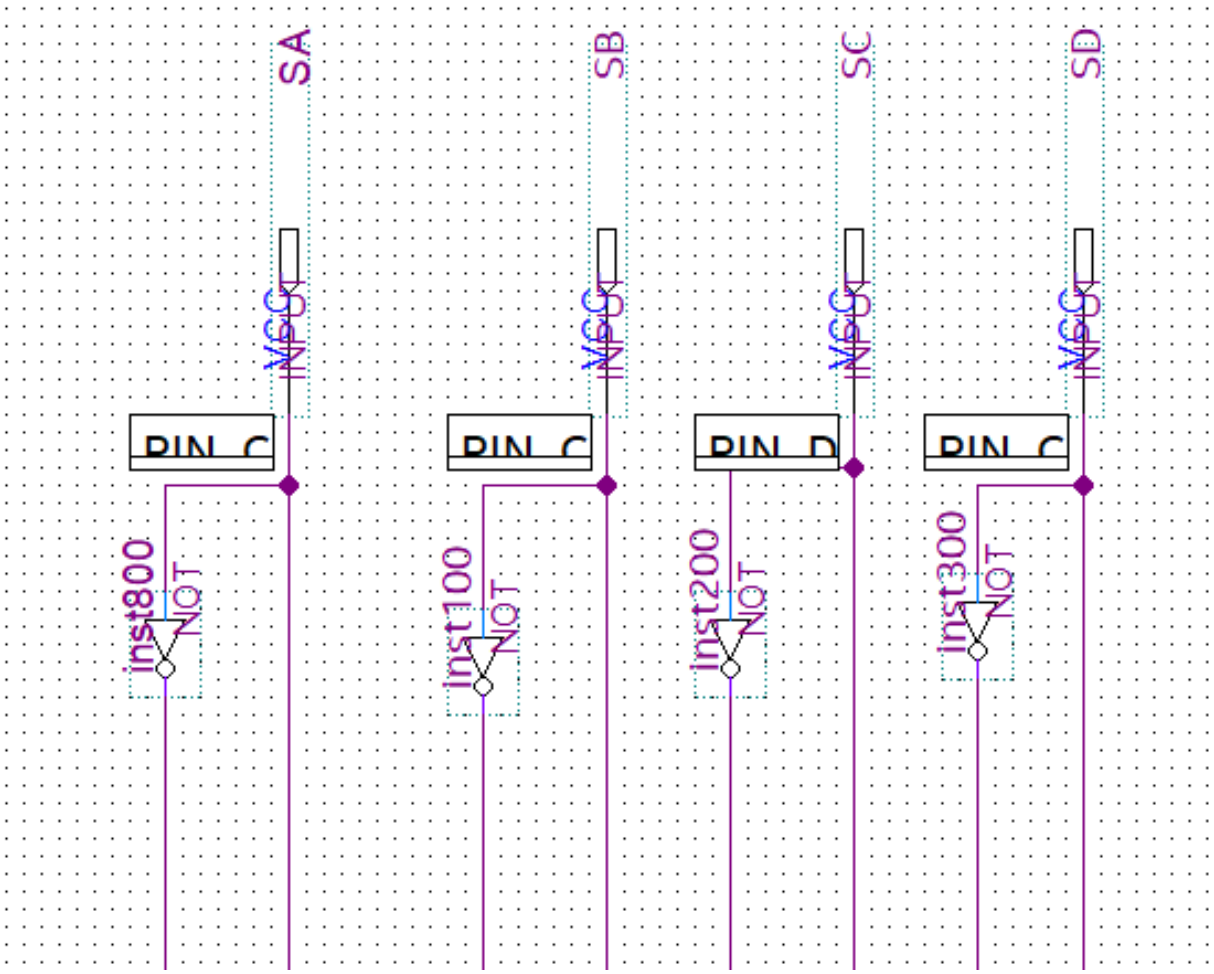

Fig. 6.1: Inputs

This shows the different inputs for the decoder, each with a true and false version. These inputs are mapped to switches, one per input.

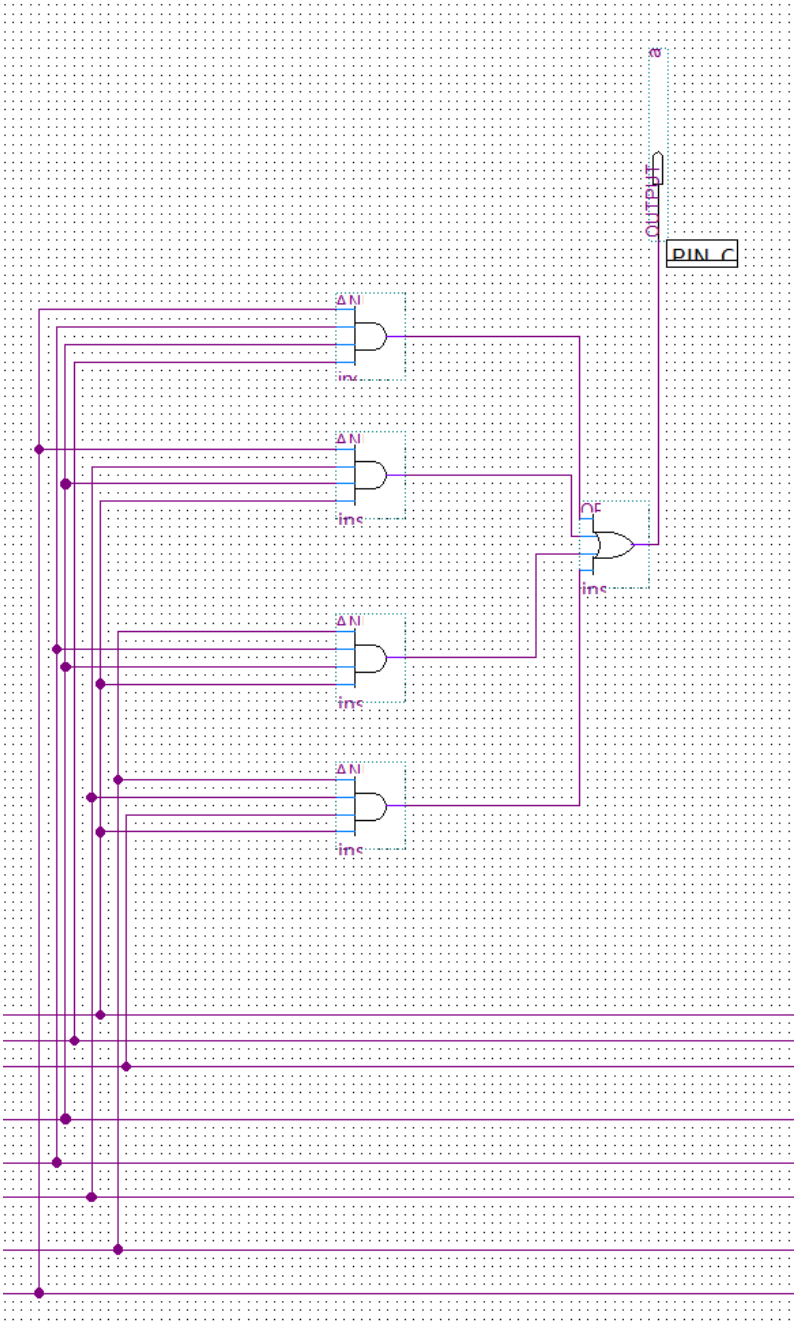

Fig. 6.2: Segment A

Segment A is the part of the decoder devoted to the output of the first segment, which is based off the first Boolean equation. It contains four AND4 gates and one OR4 gate.

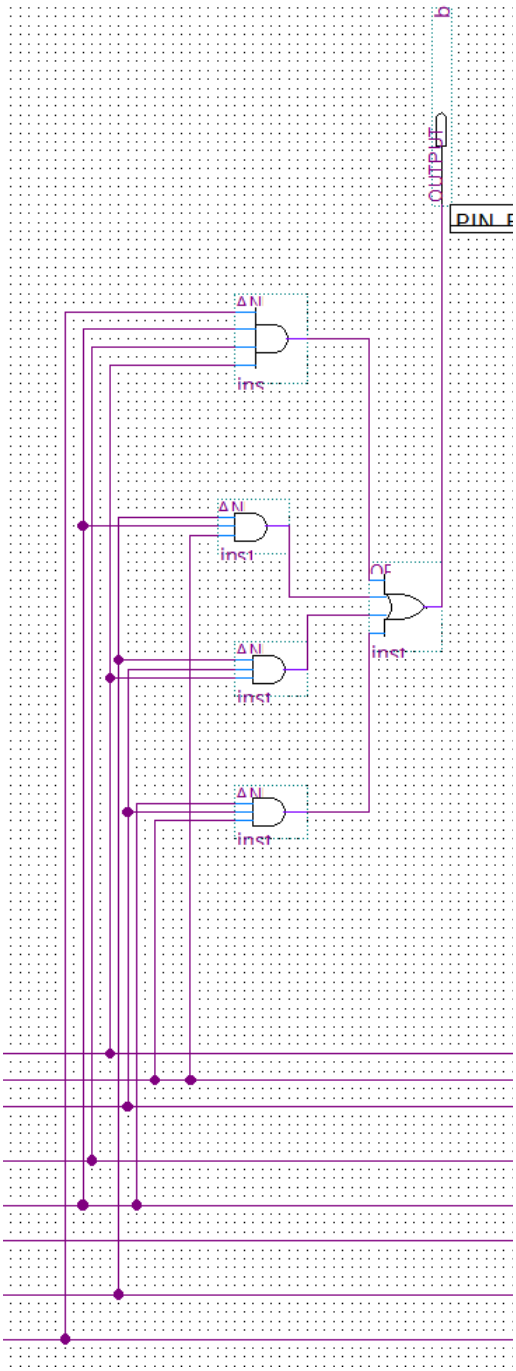

Fig. 6.3: Segment B

Segment B is the part of the decoder devoted to the output of the second segment, which is based off the second Boolean equation. It contains one AND4 gate, three AND3 gates, and one OR4 gate.

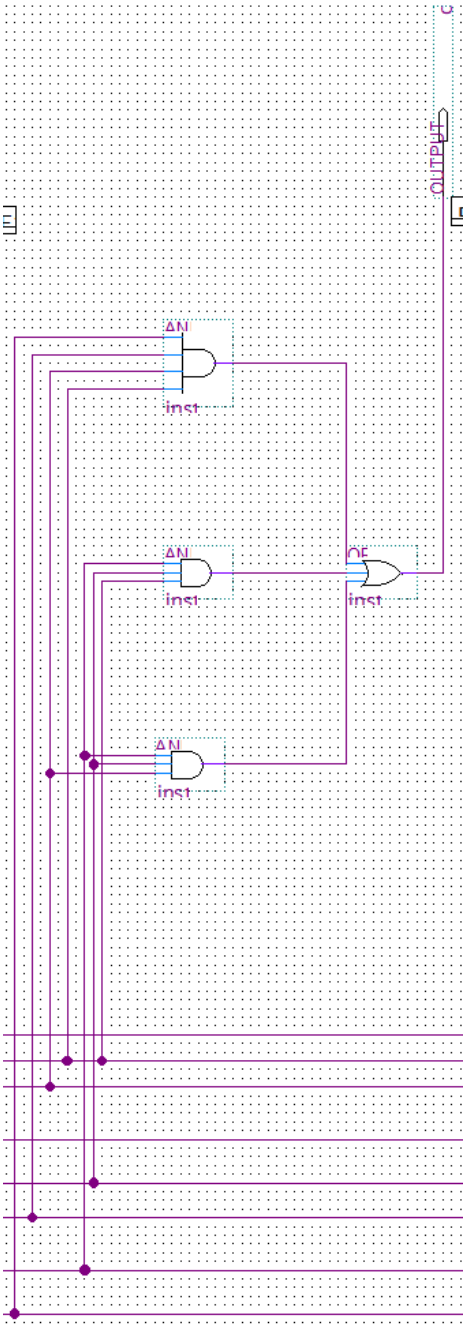

Fig. 6.4: Segment C

Segment C is the part of the decoder devoted to the output of the third segment, which is based off the third Boolean equation. It contains one AND4 gate, two AND3 gates, and one OR3 gate.

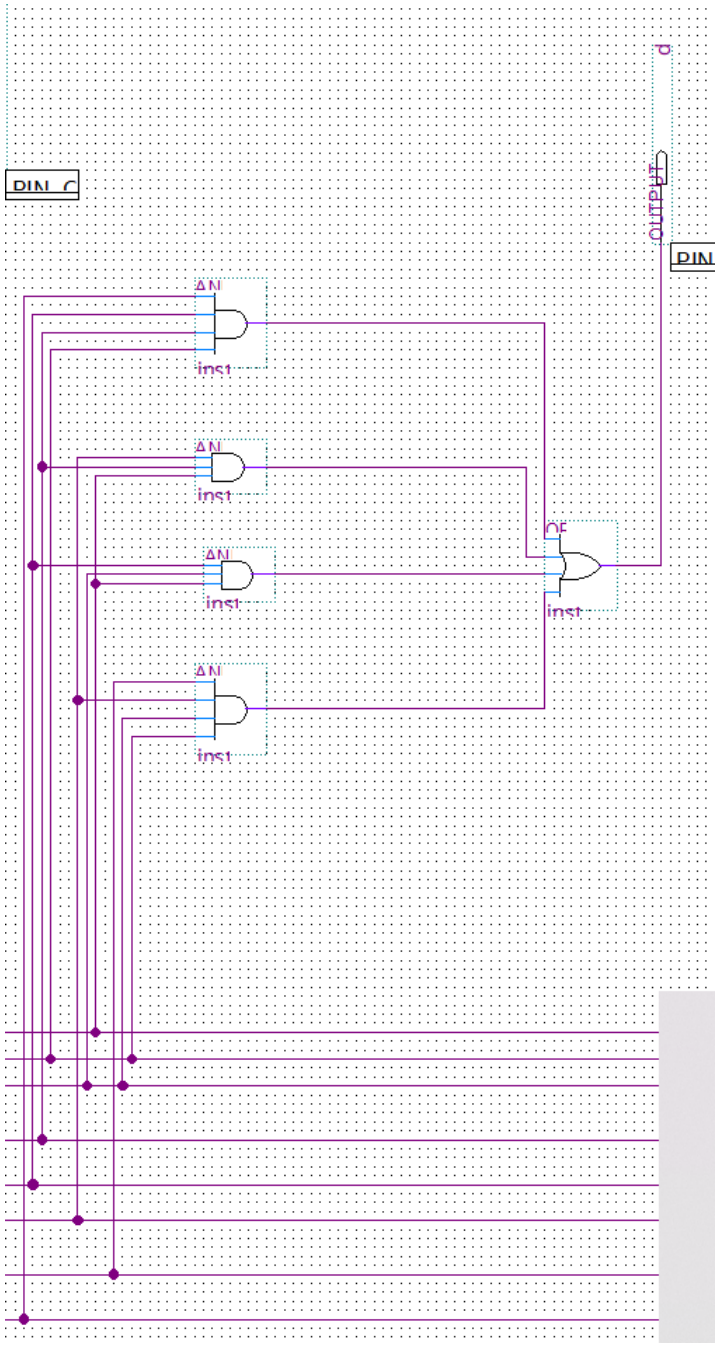

Fig. 6.5: Segment D

Segment D is the part of the decoder devoted to the output of the fourth segment, which is based off the fourth Boolean equation. It contains two AND4 gates, two AND3 gates, and one OR4 gate.

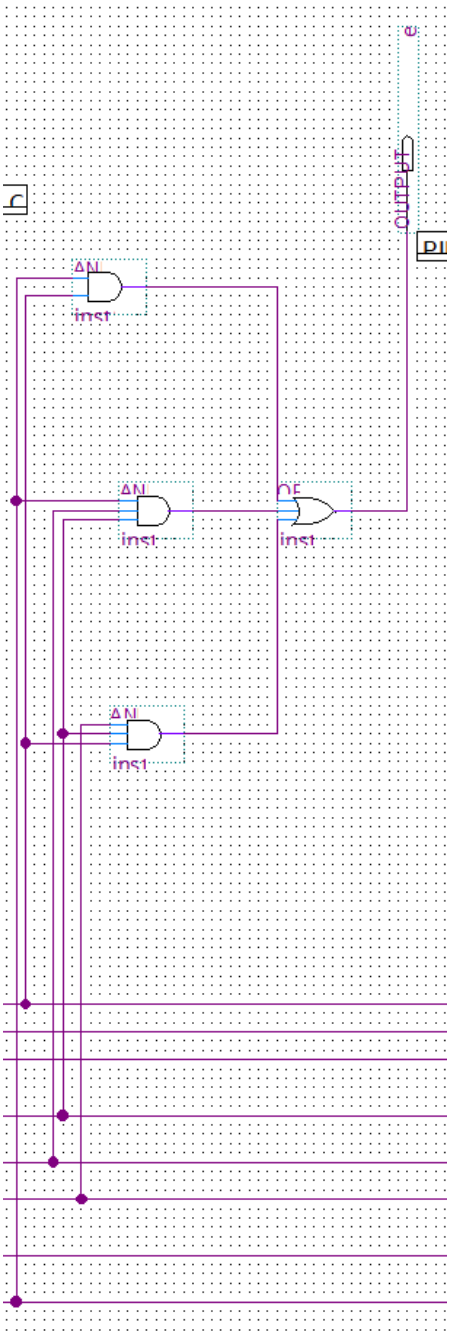

Fig. 6.6: Segment E

Segment E is the part of the decoder devoted to the output of the fifth segment, which is based off the fifth Boolean equation. It contains one AND2 gate, two AND3 gates, and one OR3 gate.

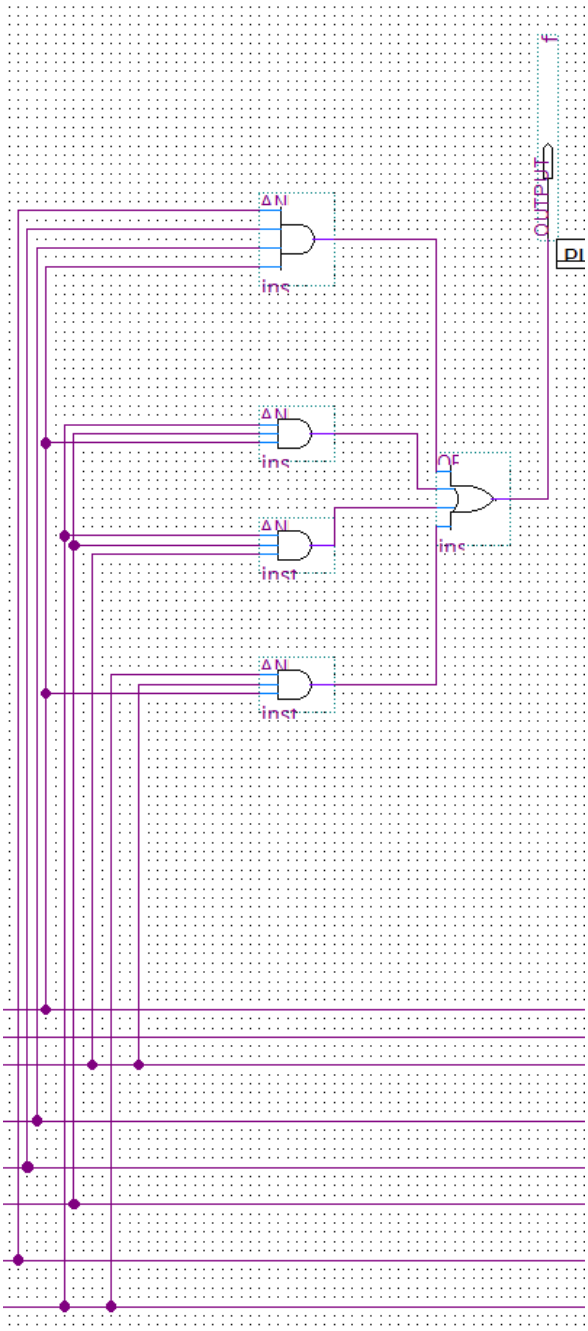

Fig. 6.7: Segment F

Segment F is the part of the decoder devoted to the output of the sixth segment, which is based off the sixth Boolean equation. It contains one AND4 gate, three AND3 gates, and one OR4 gate.

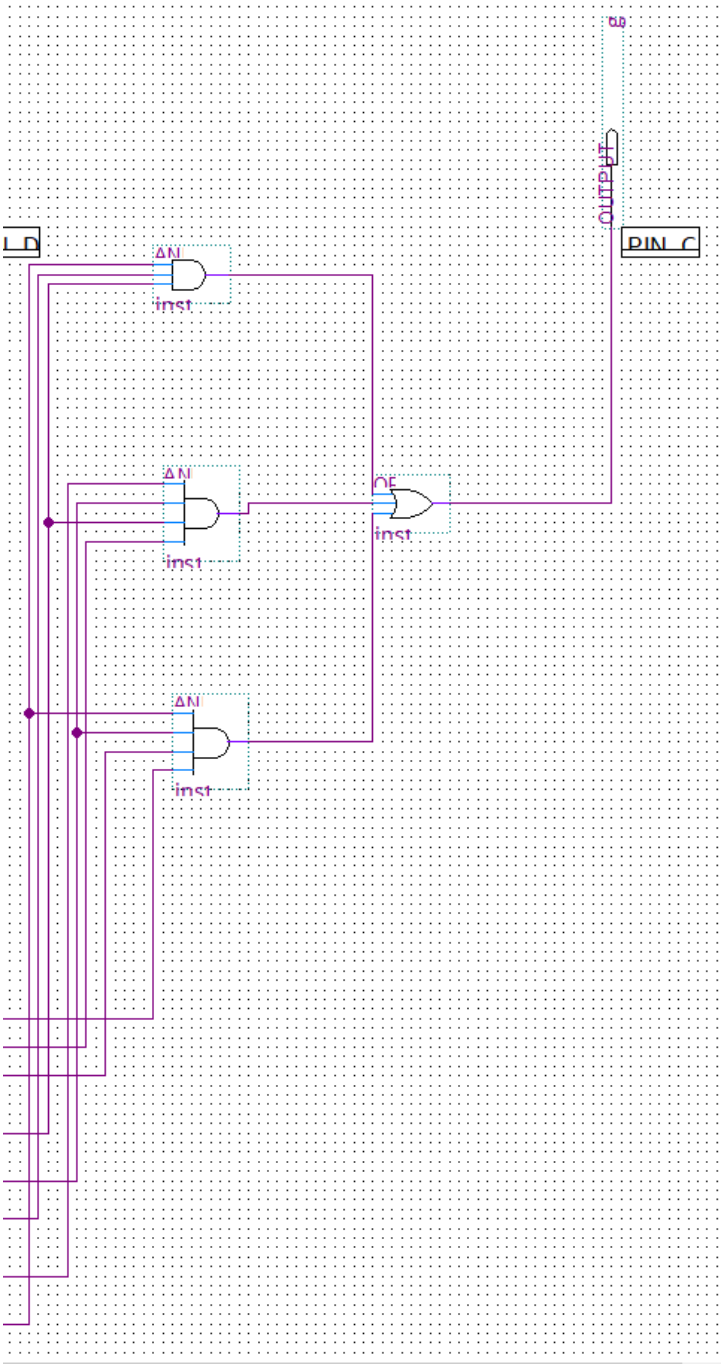

Fig. 6.8: Segment G

Segment G is the part of the decoder devoted to the output of the seventh segment, which is based off the seventh Boolean equation. It contains two AND4 gates, one AND3 gates, and one OR3 gate.

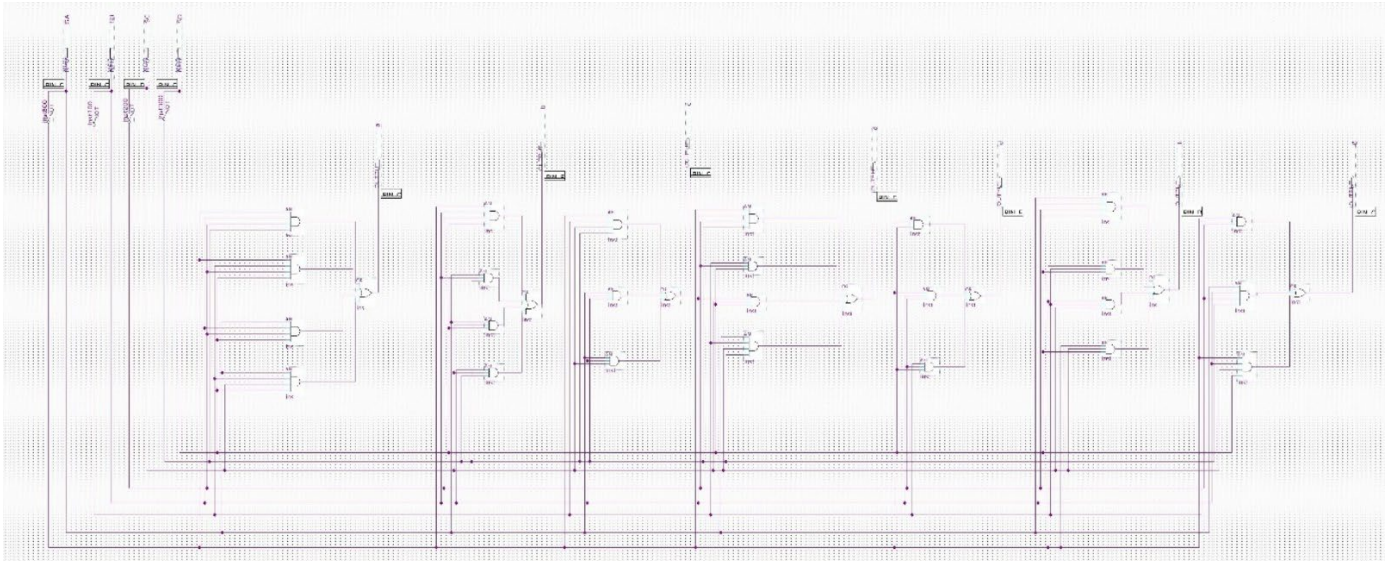

Fig. 6.9: Decoder Schematic

This is the entire schematic, which is difficult to read. That is why I captured and explained each individual segment. This schematic is the result of each individual Boolean equation converted to a separate schematic then pieced together.

Finally, I mapped each output and input to a pin on the FPGA.

Interface Name	Interface Purpose	Input or Output	Number of Pins	Active High or Low	Board Label	FPGA Pin
S_A	Gate Input	Input	1	High	Switch 0	C10
S_B	Gate Input	Input	1	High	Switch 1	C11
S_C	Gate Input	Input	1	High	Switch 2	D12
S_D	Gate Input	Input	1	High	Switch 3	C12
A	Gate Output	Output	1	Low	7SEG 0[0]	C14
B	Gate Output	Output	1	Low	7SEG 0[1]	E15

C	Gate Output	Output	1	Low	7SEG 0[2]	C15
D	Gate Output	Output	1	Low	7SEG 0[3]	C16
E	Gate Output	Output	1	Low	7SEG 0[4]	E16
F	Gate Output	Output	1	Low	7SEG 0[5]	D17
G	Gate Output	Output	1	Low	7SEG 0[6]	C17

Fig. 7: Interface Definition

This shows the pin names of each input and output as well as the number of pins, whether they are active high or low, and what their function is.

3. Results

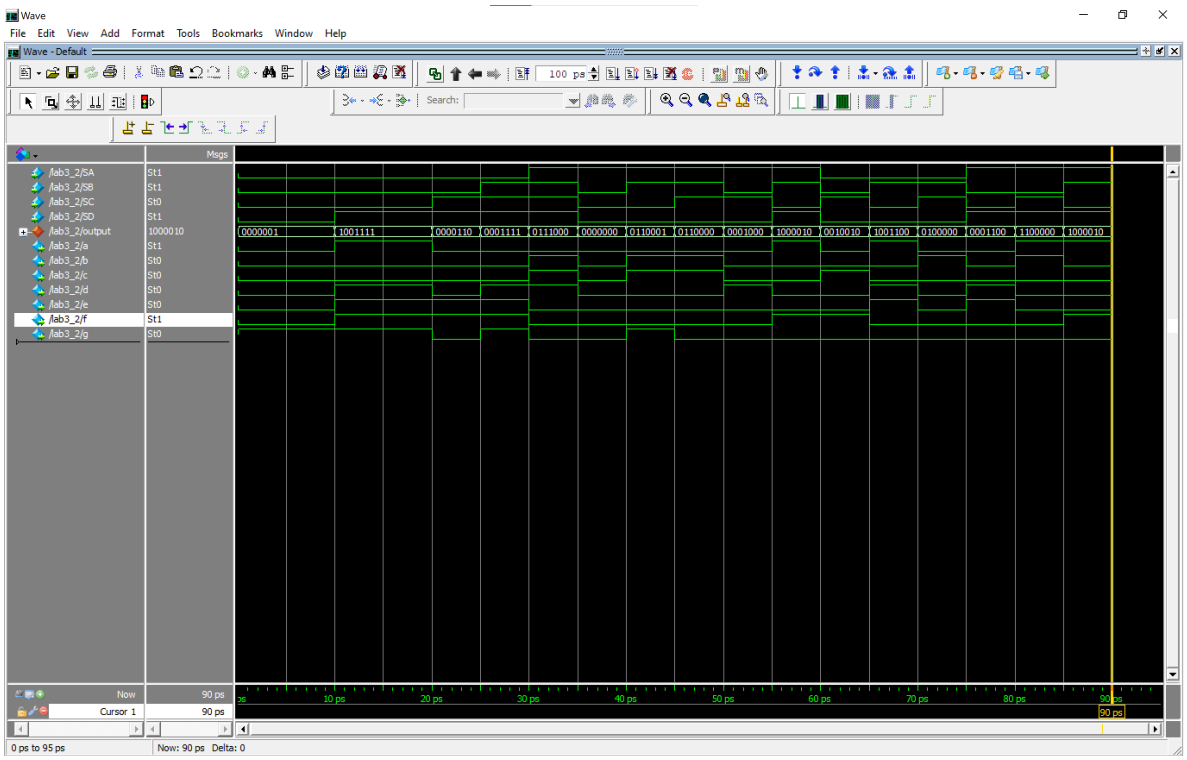

Fig. 8: Simulation

This is the final simulation I ran. The values are intended, and one screenshot is unable to convey the amount of pain I went through to get to this point. I ran the simulation last, as it was easier to test and interpret the results on the hardware.

After finalizing my design and a few back-and-forth efforts between the hardware and the schematic, I ran it on the hardware and got the expected output depicted in Fig. 1.

4. Experiment Notes

This lab did not go well for me. It took me the entire two weeks to complete, plus one day. First my Boolean equations were wrong, then the schematic was wrong, then the pins were not mapped correctly, then I forgot my FPGA at home. If anything, this was more an experiment around Murphy's Law rather than testing a schematic on an FPGA. Because of my schematic's confusing layout, I had trouble determining which inputs were mapped to which gates. But my previous unsteady Karnaugh Map and Boolean equation knowledge is now rock-steady, and I know to modulate my schematics going forward.

When I finally got this to work, I was ecstatic. It was a day late, but I was happy to have finished it by then. The only thing that went well for this lab was that I got it done eventually. Other than that, anything that could go wrong, did.

5. Study Questions

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

A: I would say a simulation is necessary whenever you are at an actual risk for damaging hardware with your design. If no one or thing is in danger, then a simulation is neither necessary nor useful. Because this lab was not at a risk for damaging the FPGA, I would argue that the only benefit of a required simulated output was that I learned the simulation software.

One of my current robotics projects is centered around simulated design. So I would say this lab was useful in that it gave me an idea of what current simulation software for digital logic is like.