Design Project 1: Design and Implementation of Combinational Circuits Utilizing LED Displays

---

Jamahl Savage

ECE 2504: Introduction to Computer Engineering

September 23, 2018

Purpose

The purpose of this project was to learn how to design, simulate, and implement a multiple – output combinational logic circuit from a given project specification. During this design process, there will be opportunities to witness how different design approaches can yield the same results. Specifically, this project will test the designer's ability to use different four-bit values inputs to design a digital circuit that will supply seven-bit values as outputs. This project also contains don't care conditions which could affect the results if the designer does not group them in Karnaugh map correctly. Depending if the digital circuit was implemented correctly, the output values will cause a LED display to show decimal digits using a DE0 Nano Board.

Technical Approach

As I thought about how to approach this project, I noticed that the decimal inputs were encoded in the (5, 3, 2, -1) format. I also noticed that there were multiple cases in which certain input combinations would result in repetitive output values or, in the case of 0001, the output is invalid. These occurrences, that were not otherwise specified to be valid input combinations, would be marked as don't care conditions. This would result in there only being nine valid and six invalid input combinations for this design. In the truth table I used to begin my design, I have marked all the outputs for the don't care conditions with the letter "d" (Figure 1). It was also important to note that an input of logic-0 lit a segment, while an input of logic-1 causes a segment to remain unlit. After I had created the truth table for this design, I created Karnaugh maps for each of the outputs, which can be seen in Appendix B. I used these Karnaugh maps to derive the minimal logic equations that would be used to create my circuit design. Instead of grouping the zeros because they would light a segment, I decided to continue forming groups of ones like normal. I chose this method because the inputs of logic-1 are still technically activating the segment to be off. I made sure to group any don't care conditions with ones if it would lead to the logic equation being optimally minimalized.

Using the minimal logic equations derived from the Karnaugh maps, I proceeded to creating my gate-level circuit. Even though the specifications state to only use 2-input NAND gates and inverters, these were still probably the best gates to use. NAND gate is a universal gate and requires few transistors to operate; therefore, NAND gates are the best to use for efficiency and cost. As seen in Figure 10, I decided to go with a design approach that took propagation delay into account. I directly connected every segment's initial inputs to the input signal wires and its complements instead of using the output of a logic gate to drive many other gates (branching out), which as a result increases the propagation delay. Thus, the propagation delay would be larger for a branching approach than the method I went with. However, I believe my method also increases the gate count of the design compared to the branching method.

To create an AND gate using only NAND gates and inverters, I simply added an inverter to the end of the NAND gate. The reason for this is that a NAND gate is just an

AND gate with its output inverted; therefore, inverting the output of an NAND gate will give the result of a regular AND gate. To create the OR gate using NAND gates, I just placed inverters on the inputs before they enter the NAND gate. The reasoning behind this is because a NAND gate is like an OR gate but with its input inverted. For the NAND gate to operate like an OR gate, inverters were placed on the inputs so that NAND gate's inversion of the inputs would be negated. However, as can be seen in Figure 10, there are no inverters present between the AND and OR gates made of NAND gates. I got rid of these inverters because to invert an output twice would only result in the original output before the double inversion. By getting rid of the extra inverters I have also effectively decreased the gate count as well.

Results

After completing the design for the gate-level circuit, I ran a simulation to verify that my circuit's behavior is consistent with my truth table (Figure 1) and the project requirements especially. As shown in Figure 9, the simulation's results are identical to the results from the truth table (Figure 1), which also matches the project requirements. In the simulation results you can also see the that the don't care cases have been labeled and contain random values that we should not acknowledge. After verifying that simulation meets the project specifications, I proceeded to implementing the gate-level circuit onto my DE0 Nano Board. According to the project specifications, I had to connect pins from the GPIO Expansion Header to the pins of my seven-segment LED display for implementation. Each pin used on the board controlled either a segment of the LED or supplied power for the LED to be able to stay lit. If I ever tried removing the connection of VCC_SYS, then the whole LED would turn off. However, if I removed the connection of a segment, it would only turn itself off and not the other segments. After setting up the connections correctly, I could verify that my circuit works by testing the valid input combinations and watching the LED display take on the value that corresponded to the input combinations I applied using the switches.

Conclusion

During the design process, I was able to analyze the possible consequences of different design approaches. While a branching approach could lower the gate count, it could also increase the propagation delay, which could be a problem if someone wanted the circuit to operate as fast as possible. Through this design project, I was able to see why the NAND gate is known as a universal logic gate. I was able to create three to four input AND and OR gates only utilizing NAND gates and inverters. I was also able to apply my knowledge about Boolean logic to simplify some of the logic gates, which as a result reduced the gate count as well. Based on these observations, it is possible to say that I was able to successfully design a functioning and efficient gate-level circuit that can drive a seven-segment LED display.

Appendix A: Truth Table

A	B	C	D	T	U	V	W	X	Y	Z
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**
0	0	0	1	d	d	d	d	d	d	d
0	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**
0	**0**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**
0	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**
0	1	0	1	d	d	d	d	d	d	d
0	1	1	0	d	d	d	d	d	d	d
0	**1**	**1**	**1**	**1**	**0**	**0**	**1**	**1**	**0**	**0**
1	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**0**	**0**
1	0	0	1	d	d	d	d	d	d	d
1	0	1	0	d	d	d	d	d	d	d
1	**0**	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**
1	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
1	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**
1	1	1	0	d	d	d	d	d	d	d
1	**1**	**1**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**

Figure 1: The truth table used to show all possible input combinations and the corresponding output for each driver circuit. Don't care conditions are non-bolded and have "d" listed for each output.

Appendix B: Karnaugh Maps

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

T = A'D

Figure 2: The Karnaugh map for segment T

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

U = AB'

Figure 3: The Karnaugh map for segment U

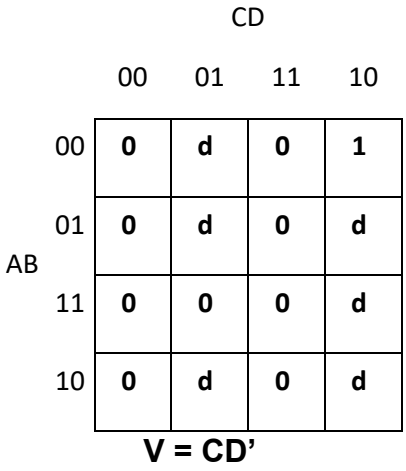

Figure 4: The Karnaugh map for segment V

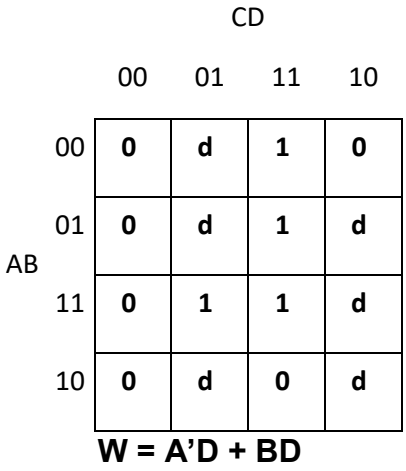

Figure 5: The Karnaugh map for segment W

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

X = A'D + A'B + BD + AB'C'

Figure 6: The Karnaugh map for segment X

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

Y = C'D + A'B'C + A'BC'

Figure 7: The Karnaugh map for segment Y

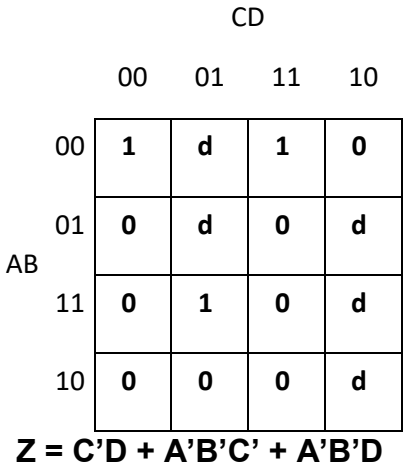

Figure 8: The Karnaugh map for segment Z

Appendix C: Simulation Results

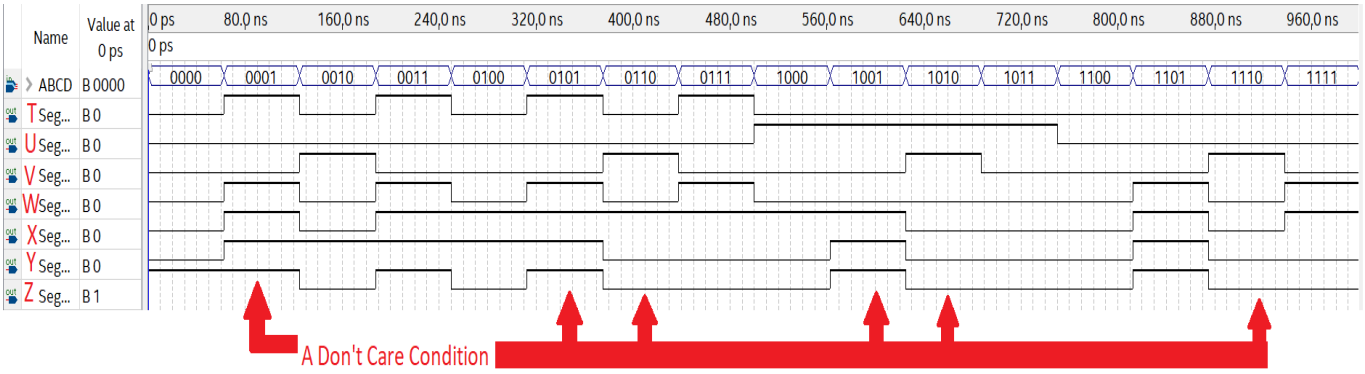

Figure 9: The simulation results of the gate-level circuit

Appendix D: Circuit Schematics

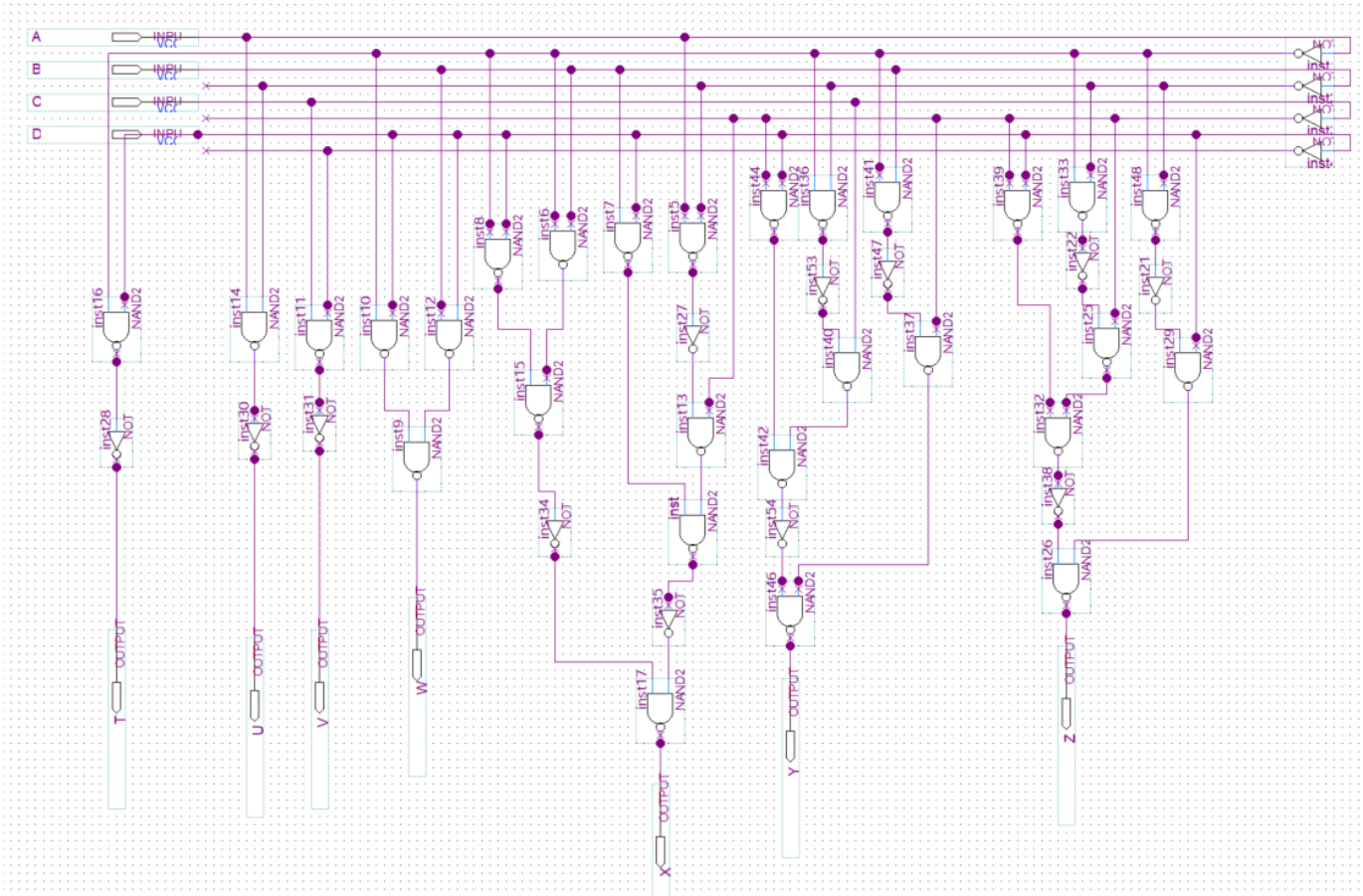

Figure 10: The gate-level circuit design used for this project