ECE 2504: Introduction to Computer Engineering
Design Project 1: Design and Implementation of Combinational Circuits

Student Name: Bowei Zhao

Honor Code Pledge: I have neither given nor received unauthorized assistance on this assignment.

Bowei Zhao

Grading: The design project will be graded on a 100 point basis, as shown below:

Manner of Presentation (30 points)

_____ Completed Cover Sheet (5 points)

_____ Organization: Clear, concise presentation of content; Use of appropriate, well-organized sections (15 points)

_____ Mechanics: Spelling and Grammar (10 points)

Technical Merit (70 points)

_____ Design Procedure Discussion (10 points)

_____ Implementation Discussion: Choice of segments and reasons for choosing them. (5 points)

_____ Supporting Figures: Truth Tables, Karnaugh maps, Circuit and Chip Layout Diagrams, etc. (15 points)

_____ Simulation Results (15 points)

_____ Validation Sheet (25 points)

_____ **Extra Credit**: Correctly implementing more than three working segments (up to 5 points)

_____ **Project Grade**

Abstract

This project is to implement a series of NAND gates and inverter chips on an A&D (‘ANDY’) board to output a alphabetical value onto an LED. The project was to be implemented by using a binary-coded-decimals which were then to be implemented in a specifically ordered karnaugh map that would give a specific gate combinational output for a specific alphabetical section of the LED display, of which together, would formulate the letter. In properly minimizing the karnaugh maps such that one would only use at most three NAND chips, we could then formulate an output such that with our twelve available NAND gates (four on each chip) would create an output. The results were that with the implementation of the ‘t’, ‘w’, and ‘x’ gates – granting that they utilized two groupings and like terms – that we could obtain an easier set of logic functions that would produce the three ‘parts’ of the whole letter we were to therefore output.

Purpose

The purpose of this project was to layout, design, implement, and craft a circuit capable of outputting an alphabetical number given a four bit binary-coded-decimal input. The project was to allow for hands-on experience in developing these schematics and working through it to get something of value.

Technical Approach

We were initially met with the objective of being given three NAND chips (with four NAND gates in it), and only one Inverter chip (with six inverters built in) to implement a circuit where an alphabetic value was to be outputted on an LED chip. We were to, using four input BCD, derive a logical truth table of what values of the corresponding sections of that LED chip (t, u v, w, x, y, z) would give us the shapes of letters A through J.

To begin, with an understanding that we were using excess-3 BCD, it was trivial that we were to implement a truth table such that all values would begin at decimal corresponding value 3 which corresponded to binary values of 0011(now decimal 0) and to end at value 12(now decimal 9) of 1100. The three initial and three ending values are thrown out due to them not representing

single ‘digits’ and also because the code is in excess-3 where you move up the binary representation of the ‘number’ by three. Table 1 shows this truth table below.

To demonstrate one sample, for example, to output a letter of A (while referring to figure 1 below), you will note that you require the segment of corresponding letter W (hereby just referred to as Segment plus a ‘Letter’) to be turned off while all the others are to be kept one when you input an Excess-3 BCD value (hereby just shorted to BCD for simplicities sake) of 0011. The project document outlines that a ‘value’ of 0 given to the segment drivers (noting that this value is different from the input BCD values) will turn that segment on while a value of 1 will turn it off. This means that segment W will have been given a value of 1 while the rest of the segment drivers T, U, V, X, Y, Z were given values of 0. Thus, for our truth tables, we fill out, for value 3 correlating to BCD value of 0011 that will output a look alike alphabetical letter A, a set of Boolean operator outputs that in this sample case would be 0 0 0 1 0 0 0. This idea formulation is constant for the rest of the ten other alphabetical letter representations that the segment display will output. This was merely one sample to demonstrate the type of processes that I went through to formulate my truth table and final set of minterms for each function.

While referring to Figure 2 below with the finished karnaugh maps and the simplied functions (noting that they have been maximized despite some digital image groupings not seeming so), and where we are also implementing the excess terms as ‘Don’t Cares’, we will arrive at a set of unique outputs for a specific input. They are:

T(A,B,C,D) = BD’+AC

U(A,B,C,D) = B’D’C’+A’C’+BD

V(A,B,C,D) = B’D’C’+BD

W(A,B,C,D) = B’D’+B’C

X(A,B,C,D) = AB+AD

Y(A,B,C,D) = BCD’+AB+ACD

Z(A,B,C,D) = AB+BC’D+ACD

The implemented gates on Quartus are shown below in Figure 3. From these gates, are the specifically implemented forms of the Boolean algebra seen above. There are 29 NAND gates in the implemented version below. It is possible to reduce these by pulling leads from previous gates

that have the same output. It was not done this time around to keep Quartus looking clean and to reduce the chance of errors and time on fixing them.

Noting from figure 3, that while they are using NAND gates and inverters that the original equation was calling for the use of AND gates, inverters, and OR gates. They all had to be switched to NAND gates and inverters as that was all we were allowed to use. To do this, I utilized an equivalency chart where an AND gate is the same as an NAND gate with an inverter on its output. An OR gate then, is just an NAND gate with both its inputs carrying an inverter. There are times that these inverters will cancel when aa newly formed NAND gate from AND and one from OR are put together where the output inverter on the NAND will cancel the input inverter on the newly changed OR.

The main implementing of the gates happened after the gates output had been verified. You were to change the chip level design at this point so that you can take advantage of the inputs and outputs of each NAND or Inverter chip and utilize these separate 'gates' on each chip to get a combinational circuit that you could then test on the A&D board granting proper wiring. This is the chip level implementations as seen in figure 5.

Granting that the right output was given (see figure 6), it was decided that building the A&D board was opportune. This is because both the chip level and gate level results have approved me for my outputs being correct.

Results

The results were such that all the gates had matching were the output as seen in figure 4 where (while ignoring all the don't care laden gates) that its 0s and 1s matched exactly those of the truth tables. Taking it a step further, noting that figure 5 is the chip level implementation, and figure 6 being the results of it. Once again, all the inputs and outputs match noting that it was successful. To match the results, you would go from 0011 which is the first input, and check your 0s and 1s to see which sectors of the segments (see figure 1) would light up.

My results were comparative to the program specifications in where in regards to figure 7, it was noted that an output that chip dummy gave was the same as the BCD values due to the same ABCD and S3-S0 values we gave it and how it was wired up to be interconnected. This was interested as it showed that the logic was correct, the Boolean was done correct as well, and that it was a real world idea that it is working.

Conclusions

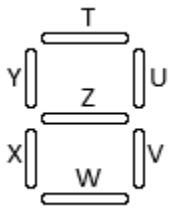
It is thus to be concluded that the project represented an exercise that allowed students to build – from the ground up – an entire logical setup that would deliver an output that is recognizable.

Personally, I felt like I learned to finally apply real world binary code, excess numbers, karnaugh maps, and Quartus into a functional real life apparatus. What I 'got' from it though, was confidence in myself to succeed and develop more of these in the future as a computer engineer and that is an invaluable trait. One observation of mine of this project would be that the Quartus software does make mistakes in how it perceives illegal wiring at times where it may just not like how you are doing it. This was a burden as my gate level implementations were at one point, extremely minimized by borrowing outputs from other gates. This was not able to be implemented due to timing constraints. Another point was that if we were given more time, utilizing six or more NAND gates to implement the full seven-segment display may have been 'better' for the students as you can actually see an output that is recognizable at that point. Having only parts of them isn't problematic as much as it is just a slightly flat climax to a long project.

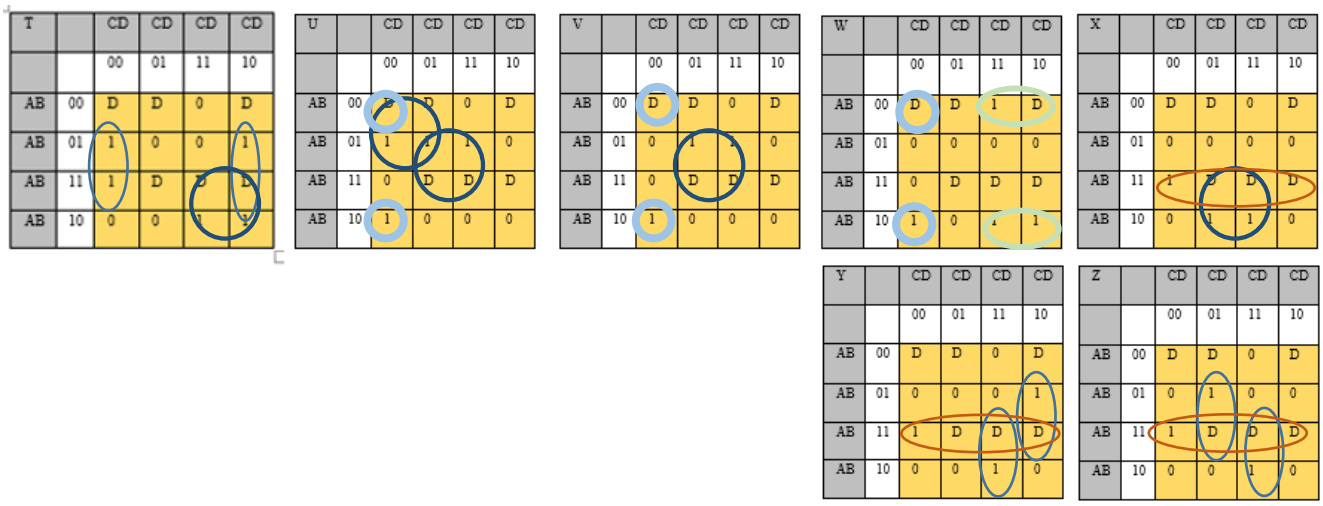
Appendicies

Value	A	B	C	D	T	U	V	W	X	Y	Z	
0	0	0	0	0	d	d	d	d	d	d	d	
1	0	0	0	1	d	d	d	d	d	d	d	
2	0	0	1	0	d	d	d	d	d	d	d	
3	0	0	1	1	0	0	0	1	0	0	0	A
4	0	1	0	0	1	1	0	0	0	0	0	B
5	0	1	0	1	0	1	1	0	0	0	1	C
6	0	1	1	0	1	0	0	0	0	1	0	D
7	0	1	1	1	0	1	1	0	0	0	0	E
8	1	0	0	0	0	1	1	1	0	0	0	F
9	1	0	0	1	0	0	0	0	1	0	0	G
10	1	0	1	0	1	0	0	1	0	0	0	H
11	1	0	1	1	1	0	0	1	1	1	1	I
12	1	1	0	0	1	0	0	0	1	1	1	J
13	1	1	0	1	d	d	d	d	d	d	d	
14	1	1	1	0	d	d	d	d	d	d	d	
15	1	1	1	1	d	d	d	d	d	d	d	

(Table 1: Truth Table for Excess-3 Boolean Algebra)



(Figure 1: Seven-Segment LED segmented drivers and letter corelations)



T		CD	CD	CD	CD
		00	01	11	10
AB	00	D	D	0	D
AB	01	1	0	0	1
AB	11	1	D	D	D
AB	10	0	0	1	1

U		CD	CD	CD	CD
		00	01	11	10
AB	00	D	D	0	D
AB	01	1	1	1	0
AB	11	0	D	D	D
AB	10	1	0	0	0

V		CD	CD	CD	CD
		00	01	11	10
AB	00	D	D	0	D
AB	01	0	1	1	0
AB	11	0	D	D	D
AB	10	1	0	0	0

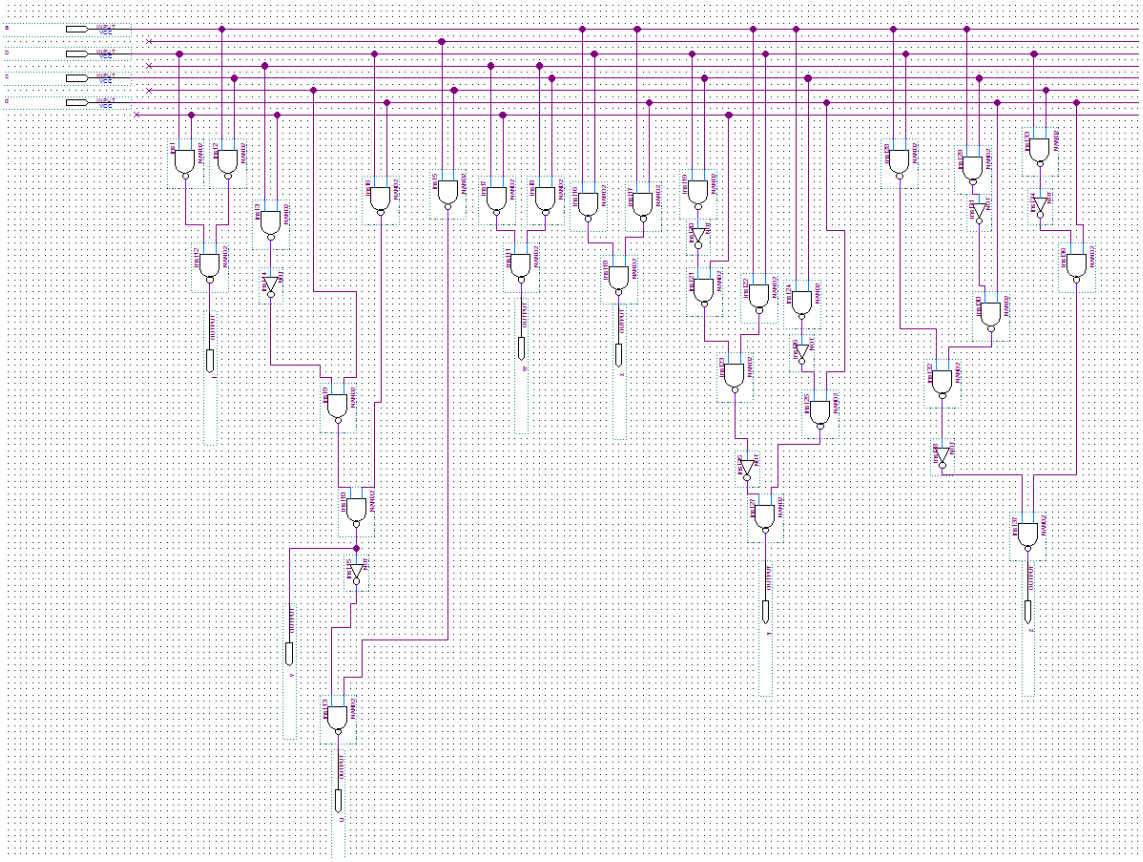
W		CD	CD	CD	CD
		00	01	11	10
AB	00	D	D	1	D
AB	01	0	0	0	0
AB	11	0	D	D	D
AB	10	1	0	1	1

X		CD	CD	CD	CD
		00	01	11	10
AB	00	D	D	0	D
AB	01	0	0	0	0
AB	11	1	D	D	D
AB	10	0	1	1	0

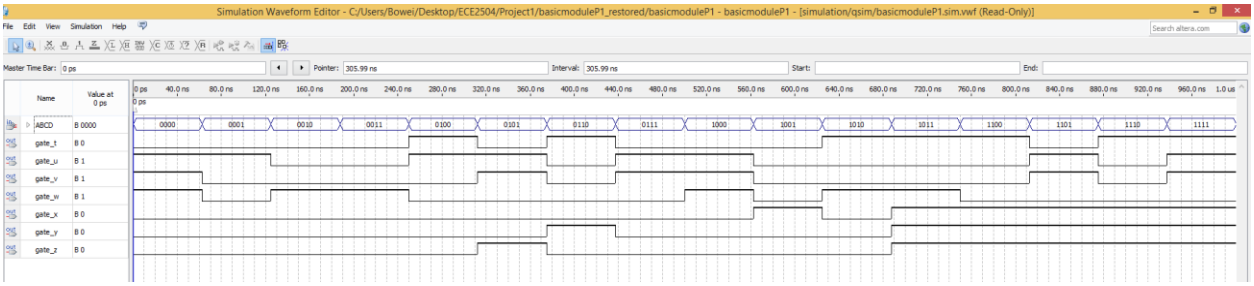
Y		CD	CD	CD	CD
		00	01	11	10
AB	00	D	D	0	D
AB	01	0	0	0	1
AB	11	1	D	D	D
AB	10	0	0	1	0

Z		CD	CD	CD	CD
		00	01	11	10
AB	00	D	D	0	D
AB	01	0	1	0	0
AB	11	1	D	D	D
AB	10	0	0	1	0

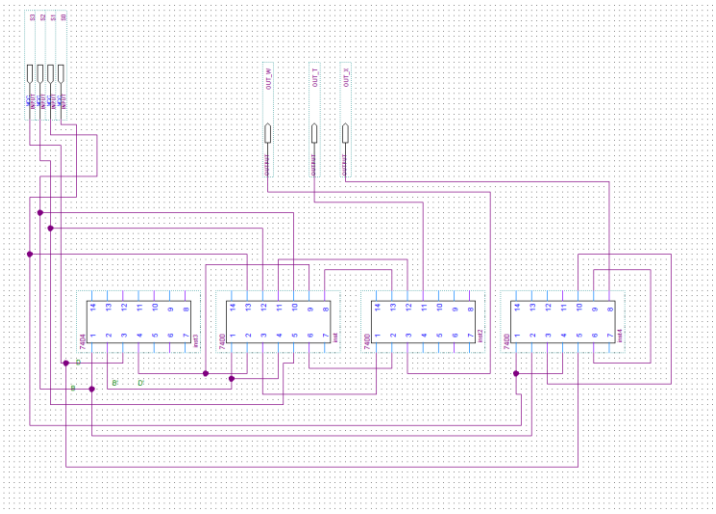
(Figure 2: Karnaugh Map Implementation)



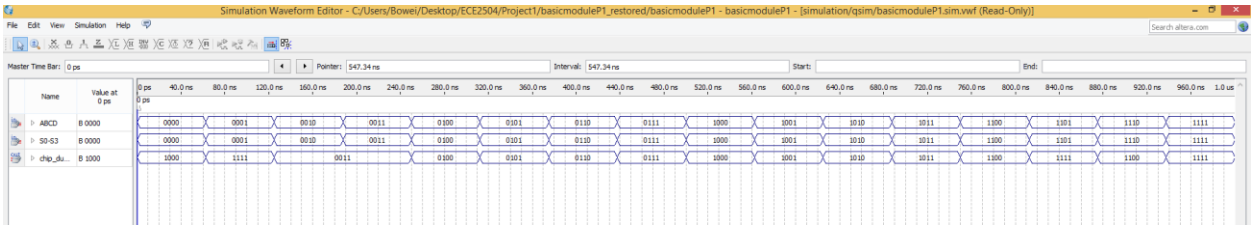
(Figure 3: Gate Implementation Diagram)



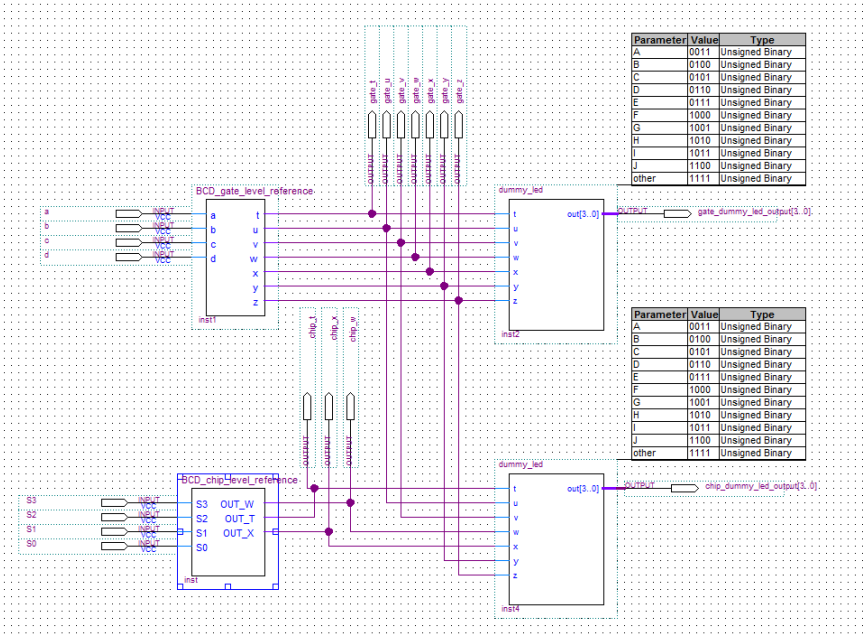
(Figure 4: Results of Gate Implementation)



(Figure 5: Chip Level Implementation)



(Figure 6: Results of Chip Level Design)



(Figure 7: Main Chip Diagram)