Benny Chen
1/25/2022
Lab 3

Theory

We used Karnaugh Maps to simplify and get an equation for each bit which then we made a logicworks prototype which is used to plan the hardware. Karnaugh Maps are used to simplify and create equations. For example, taking the D bit in XS3, we start filling in values corresponding to the POSTNET equivalent. For the first value of POSTNET which is also equivalent to 0 is 11000, and the corresponding D bit is 0 so we fill in the 0 position, or 0000, of the k-map with 0. Since there are 5 variables in POSTNET, we use the last value, V to create 2 instances with one if it's high V and another when low V'. We then fill out every value for each bit in XS3. For all the remaining empty boxes, we would put an X in them or "don't care" values. We put these X's in because we "don't care" what that value outputs as we don't ever use that input. We now have to create and simplify an equation using the k-maps. We create an equation by "grouping" together like high terms and don't care terms. By grouping them together we would simplify the equation as if a variable in that grouping has another high term then why not just have them together to simplify the circuit. We can group together X's as they don't matter as we wont ever hit those values which often makes it the equation simplifier, however we can't group together 0's as we do hit those values and would return the wrong output.

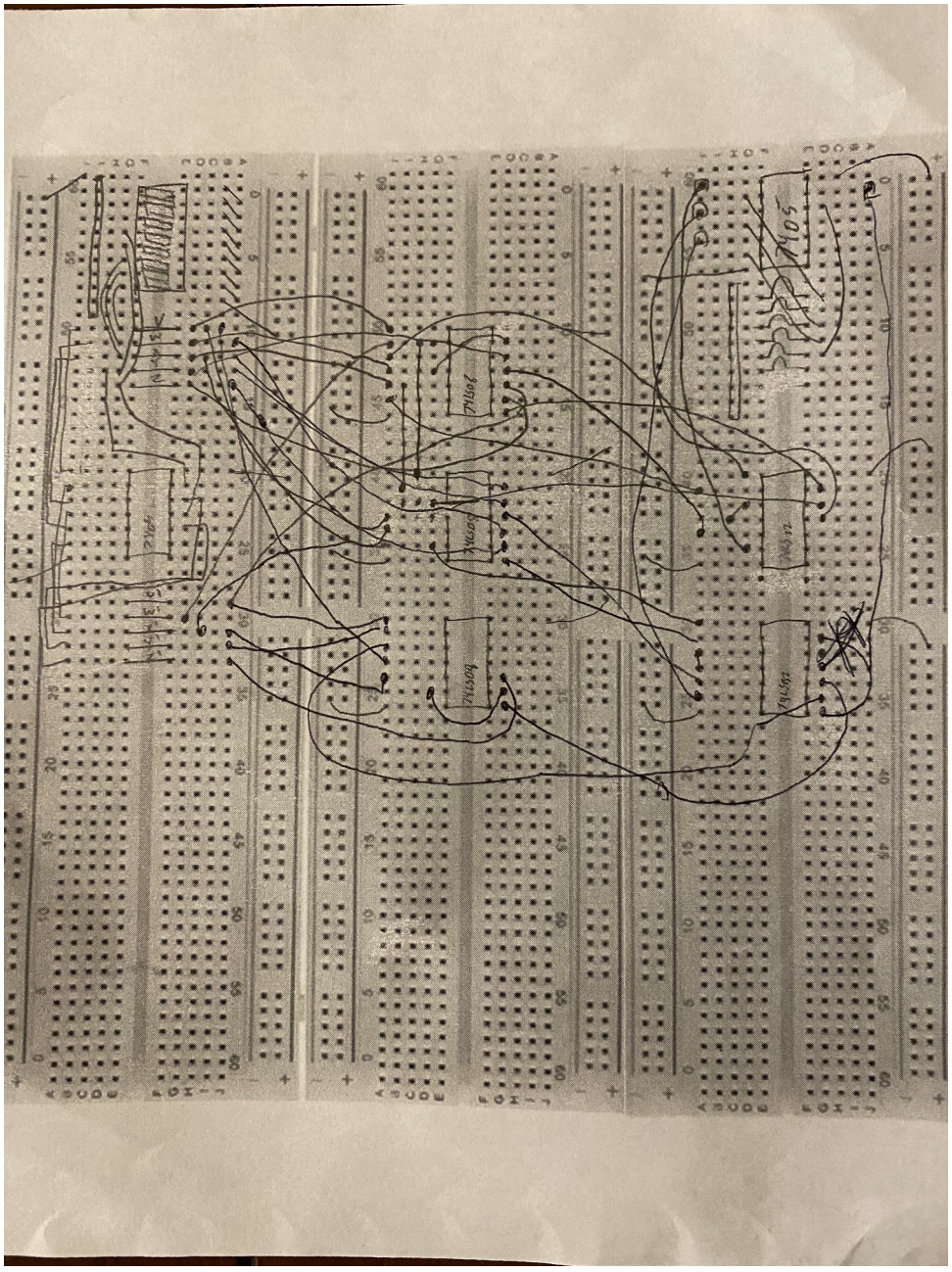
POSTNET is a numbering system often associated with the zip code of an address. There's a total of 9 numbers in a zip code, and each number is translated from decimal to POSTNET which has a weighting of 74210. POSTNET is represented by only having 2 out of the 5 states values being "high" or used. The numbers start with decimal value 0 of 11000, which is the only way to represent 0 with 7 + 4 = 11. It then jumps to 00011 as 1 and increments up from that. XS3 is the same as binary but it has an "excess of 3", so instead of binary starting at 0000, XS3 starts at 0011. The next number would then be 0100 and so on.

A invalid input code would be any code that has only 1 "high" value or more than 2 "high" values. There would then be a total of 22 invalid input codes.

Deliverables

Equations:

D: V.W’ + W.X + W.Y
C: V’.Z + W’.X
B: V.X’ + W.Z +X.Y
A: V’.Y’ + V.X’.Z’



ZIP code: 06117

	V	W	X	Y	Z

Decimal	7	4	2	1	0
0	1	1	0	0	0
6	0	1	1	0	0
1	0	0	0	1	1
1	0	0	0	1	1
7	1	0	0	0	1
0	1	1	0	0	0
0	1	1	0	0	0
0	1	1	0	0	0
1	0	0	0	1	1

POSTNET Code: 11000 01100 00011 00011 10001 11000 11000 11000 00011

Discussion

In this lab we learned about actualizing our prototypes from our logicworks and drawings to actual hardware. It was somewhat tricky with connecting everything together and I had to debug a couple times due to wrong outputs. I learned about how to manage wires to neatly connect everything and found out how useful logic probes are to debug. I also learned more about minimizing karnaugh maps which is very useful in simplifying equations.

Questions

1. **Convert unsigned binary 1011 0111(Base 2) to ternary and hexadecimal.**

Decimal: 183

Ternary: 20210

81	**27**	**9**	**3**	**1**
2/3 = .66 **.6 * 3 = 2**	**6/3 = 2** **0 * 3 = 0**	**20/3 = 6.6** **.6 * 3 = 2**	**61/3 = 20.3** **.3 * 3 = 1**	**183/3 = 61.0** **0 * 3 = 0**
2	**0**	**2**	**1**	**0**

81 * 2 = 162
27 * 0 = 0
9 * 2 = 18
3 * 1 = 3
1 * 0 = 0
162 + 0 + 18 + 3 + 0 = 183

Hex: B7

16	**1**
11/16 = .6875 | .6875 * 16 = 11	**183/16 = 11.4375 | .4375 * 16 = 7**
11 = B	7

1 * 7 = 7
16 * 11 = 176
176 + 7 = 183

2. **Convert the POSTNET (9) symbology bar code below into its decimal representation. Do not include the checksum.**

10100	00110	10001	01010	11000	11000	00101	10001	01010
9	3	7	5	0	0	2	7	5

3. **Draw a circuit using only AND-2, OR-2, and INV/NOT gates that satisfies the following formula:** $D(C'B + A') + \overline{(D + C')}$. **Don't forget about DeMorgan's theorem!**

$D(C'B+A') + \overline{(D+C')}$ ⇒ Demorgans $\overline{(A+B)} = \bar{A}\bar{B}$

$DC'B + DA' + D'C$