Andrew Lorber

Professor Risbud

DLD 2018

Abstract

The Purpose of this project was to create a circuit that would mimic a traffic light. It would include the following: a red, yellow, and green LED for all four directions; a green and yellow turn signal for East and West; a seven segment display to count the time left for a green light; a pedestrian button that would change the remaining time for a green light to five seconds. The traffic light will loop through a series of states, each a certain number of seconds long. The goal of the project was to design a functional traffic light as efficiently as possible. In order to complete the project, two plausible designs were to be made, but only one was to be built. The first step was to divide the project into separate parts: time counter, state loop, and output. Using this layout, it is easier to assess each part and derive multiple approaches to the circuit. Once different approaches have been found, the different parts of the circuit are combined in a schematic, which can then be tested and built.

Design

The twelve states are the following [0 means lit]:

State #	# Seconds	E->W Turn		N/S			W->E Turn		E			W		
X	X	G	Y	G	Y	R	G	Y	G	Y	R	G	Y	R
0	2	1	1	1	1	0	1	1	1	1	0	1	1	0
1	4	0	1	1	1	0	1	1	1	1	0	1	1	0
2	2	1	0	1	1	0	1	1	1	1	0	1	1	0
3	1	1	1	1	1	0	1	1	1	1	0	1	1	0
4	19	1	1	0	1	1	1	1	1	1	0	1	1	0
5	5	1	1	1	0	1	1	1	1	1	0	1	1	0
6	2	1	1	1	1	0	1	1	1	1	0	1	1	0
7	4	1	1	1	1	0	0	1	0	1	1	1	1	0
8	2	1	1	1	1	0	1	0	0	1	1	1	1	0
9	1	1	1	1	1	0	1	1	0	1	1	1	1	0
10	15	1	1	1	1	0	1	1	0	1	1	0	1	1
11	3	1	1	1	1	0	1	1	1	0	1	1	0	1
12	2	1	1	1	1	0	1	1	1	1	0	1	1	0

The first step is to divide the circuit into sections: the time counter, the state loop, and the output LEDs. The first issue I worked on was the state loop. After much consideration, I decided that the best approach would be to use a demultiplexer. Each output of the demux would be a different state and I would cycle through each state with a counter. The states would be changed by having the clock input of the counter be an output of logic, which would output a '1' when the given time of the state passes. An issue that arose was the fact that two demuxes were needed, therefore I had to differentiate between the two and account for the fact that there were more outputs than states. I differentiated between the two demuxes by making use of the inhibit pin. I connected the two inhibit pins to D and D' which allowed me to select which demux to use. In order to deal with the unused outputs that did not relate to a state, I used the preset pin on the counter and made the blank output(s) the last of the demux. The jam inputs for A, B, and C were set to ground and the jam input for D was set to D' so it would switch to the next demux. The input of the preset pin was delivered from logic that would output a '1' when an unused output was reached. Since the lights were required to light on an input of '0', I inputted ground into the

multiplexer. However, in order to account for the float state of the outputs that are not “chosen”, I placed a pull up resistor at each output. This assured that each non-chosen state would output a ‘1’, while each chosen state outputted a ‘0’.

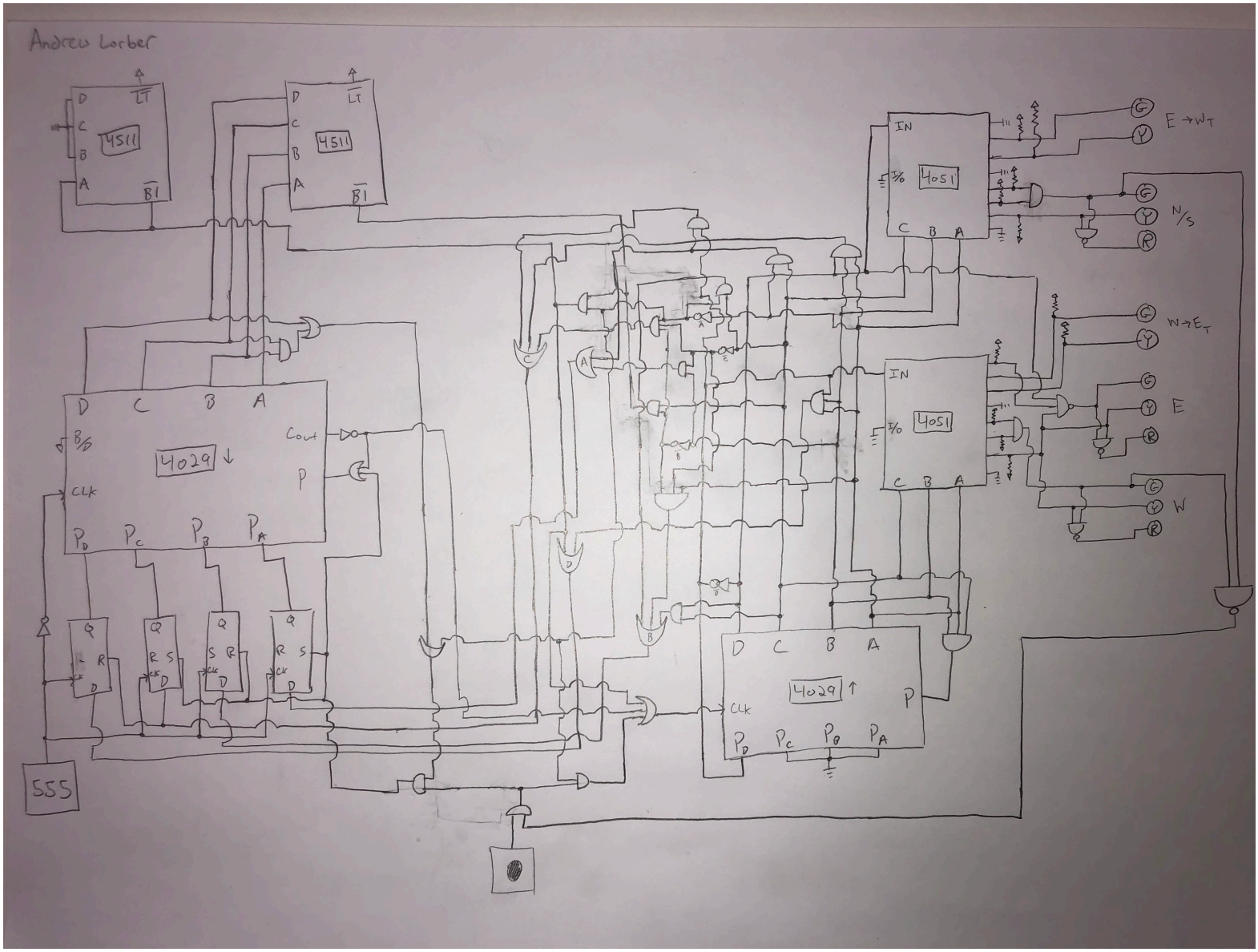
The next problem I thought about was how to keep track of the number of seconds in a given state. I decided on two possible approaches: count up to the given number or count down from the given number. Both of these methods will be discussed later.

The last step was to configure the logic to the LEDS. Since most of the yellow and green LEDs related to only one output from the demux, they were simply connected directly. However, problems arose for the red LEDs and the East green LED, since they are active for multiple states. I solved the issue with the red LEDs by having the input to them be logic from their corresponding green and yellow LEDs. I used logic, specifically NAND gates, to check if either of the corresponding green or yellow LEDs were active. If both green and yellow LEDs were inactive for a given direction then the red LED would become active. I solved the issue with the East green LED by having the input of the LED be a function of the states it is inactive during. Since it is inactive for two states on one demux and all the states on another, I was able to use a NAND gate in the same fashion as with the red LEDs.

Design #1 – Counting Down

My first design for the traffic light was to count down from a specific number. I would have a demux cycle through each state and to count the seconds in a state I would have a second counter count down from the given number. In order to count from the given numbers on a four-bit counter, I decided to split large numbers into smaller numbers that could be represented in four bits. Since the counter is also being used for the seven-segment display, I chose to split the

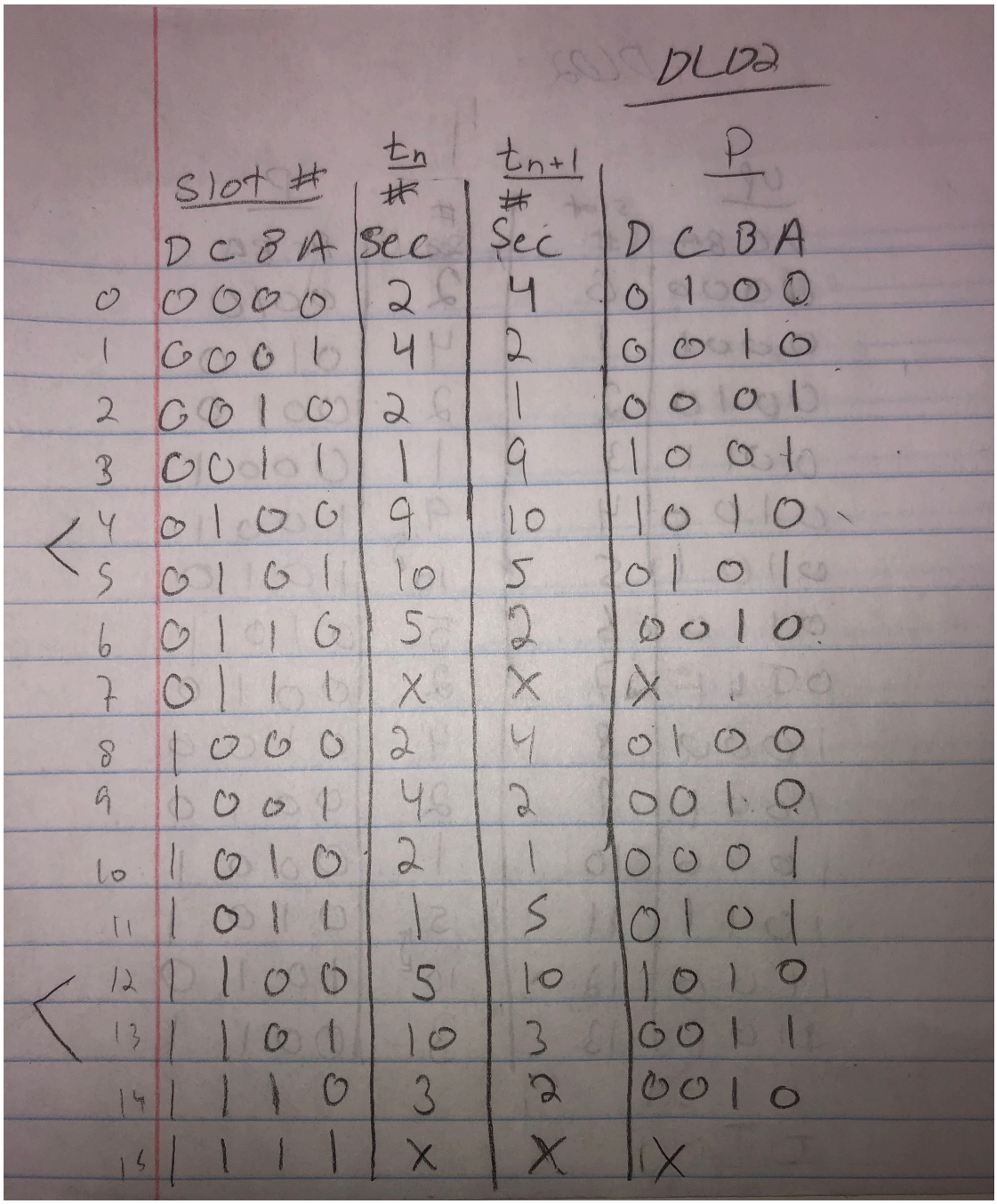
number into ten and five/nine. This made it easier to program the seven-segment display. The schematic diagram for this circuit can be seen below:



The counter works by making use of the jam pins. Logic is used to derive the necessary inputs to the jam pins from the current output of the state counter. In order to account for timing issues of non-ideal chips, I made the logic from the state counter output the jam inputs of the next state and use D type flip flops to delay. For example, the first state is two seconds and the second state is four seconds. During the first state the output from the logic will be the jam inputs

that will set the counter to four seconds. The flip-flop will then bring that input to the jam pins, so when the counter hits zero and the preset pin is activated, the counter is preset to four immediately, without any timing issues, and not what the logic now outputs.

The Karnaugh Maps for the logic described above can be seen below:



DLD2

	Slot #	t_n #	t_{n+1} #	P
	D C B A	Sec	Sec	D C B A
0	0 0 0 0	2	4	0 1 0 0
1	0 0 0 1	4	2	0 0 1 0
2	0 0 1 0	2	1	0 0 0 1
3	0 0 1 1	1	9	1 0 0 1
4	0 1 0 0	9	10	1 0 1 0
5	0 1 0 1	10	5	0 1 0 1
6	0 1 1 0	5	2	0 0 1 0
7	0 1 1 1	X	X	X
8	1 0 0 0	2	4	0 1 0 0
9	1 0 0 1	4	2	0 0 1 0
10	1 0 1 0	2	1	0 0 0 1
11	1 0 1 1	1	5	0 1 0 1
12	1 1 0 0	5	10	1 0 1 0
13	1 1 0 1	10	3	0 0 1 1
14	1 1 1 0	3	2	0 0 1 0
15	1 1 1 1	X	X	X

Andrew Lorber

J_A

BA \ DC	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	1	X	X	1
10	1	0	0	1

$Ac + B\bar{C}$

J_B

BA \ DC	00	01	11	10
00	0	1	1	0
01	1	0	1	1
11	0	X	X	0
10	0	1	1	0

$CD + \bar{A}C + A\bar{B}\bar{C}$

J_C

BA \ DC	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	0	X	X	1
10	0	0	0	0

$\bar{A}\bar{B}\bar{C} + AC\bar{D} + AB\bar{D}$

$\bar{A}\bar{B}\bar{C} + (AC)\bar{D} + AB\bar{D}$

J_D

BA \ DC	00	01	11	10
00	0	1	1	0
01	0	0	0	0
11	1	X	X	0
10	0	0	0	0

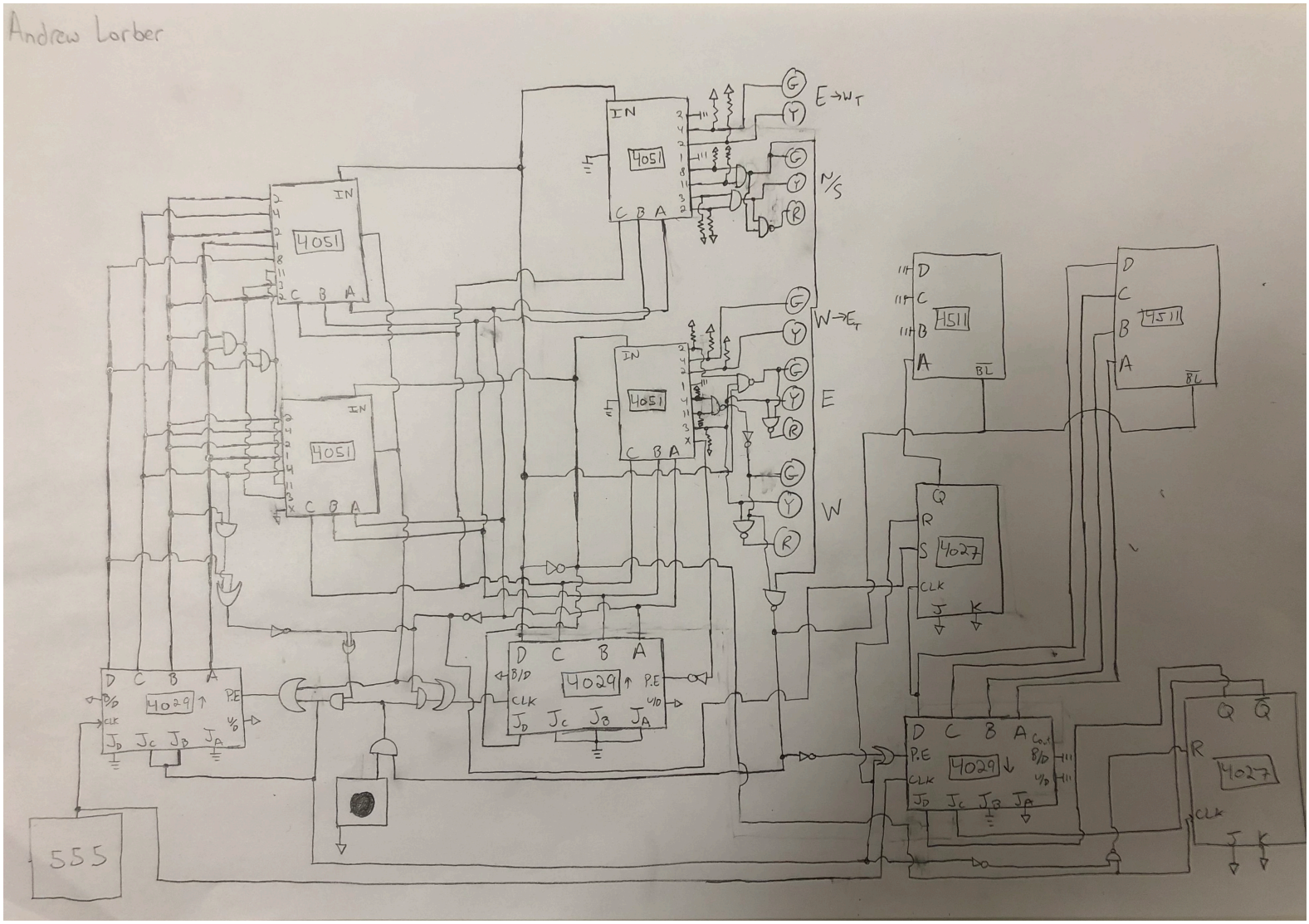
$\bar{A}\bar{B}C + AB\bar{D}$

$\bar{A}(\bar{B}C) + AB\bar{D}$

The last step in designing this circuit was the button. The button works by using the set and reset pins on the flip-flops. When the button is pressed, it uses the set and reset pins of the flip-flop to make the output to the jam pins five and activates the preset pin. However, logic is used to assure that the button will only work when the LEDs are green, to assure that the time can only be set to five if there are more than five seconds left, and to assure that if the state is split into two states due to the time being too long, the state counter will be incremented if necessary.

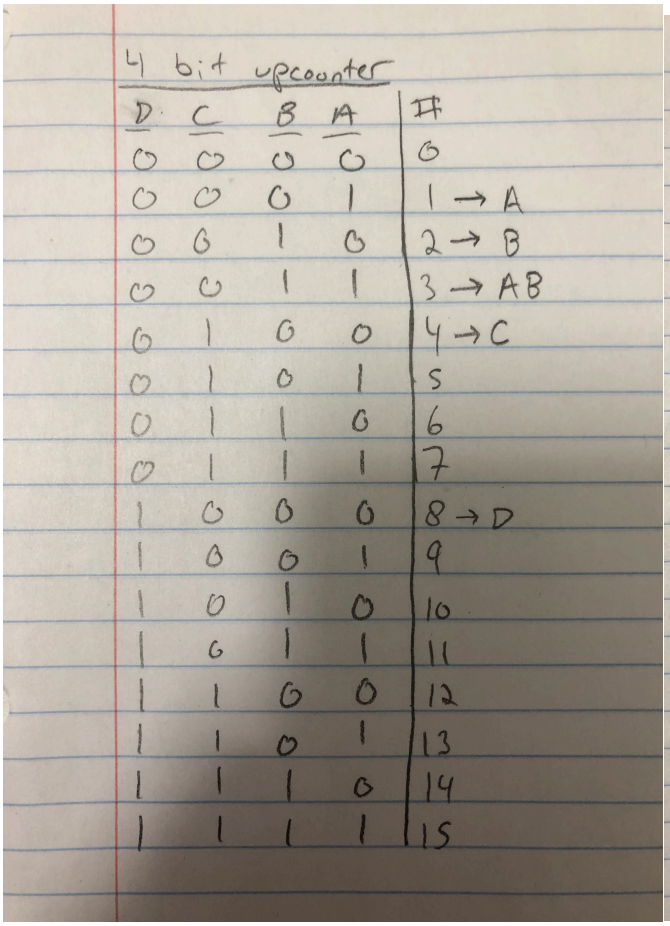
Design #2 – Counting Up (Chosen Design)

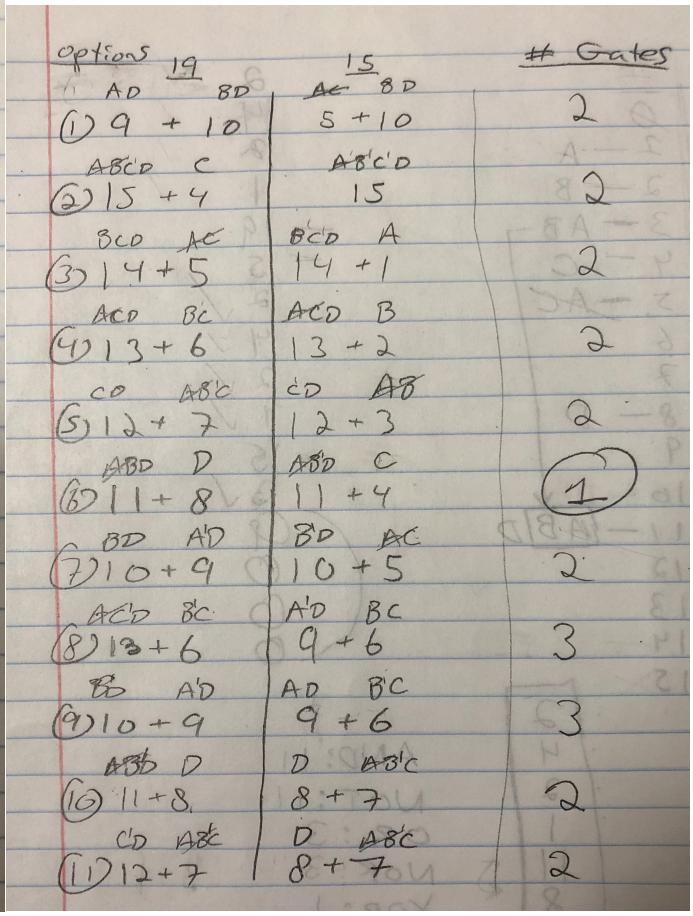
My second design is different to the first in that it counts up to the given number when timing states. The circuit works by using a multiplexer and logic to time states. A schematic diagram of the circuit can be seen below:

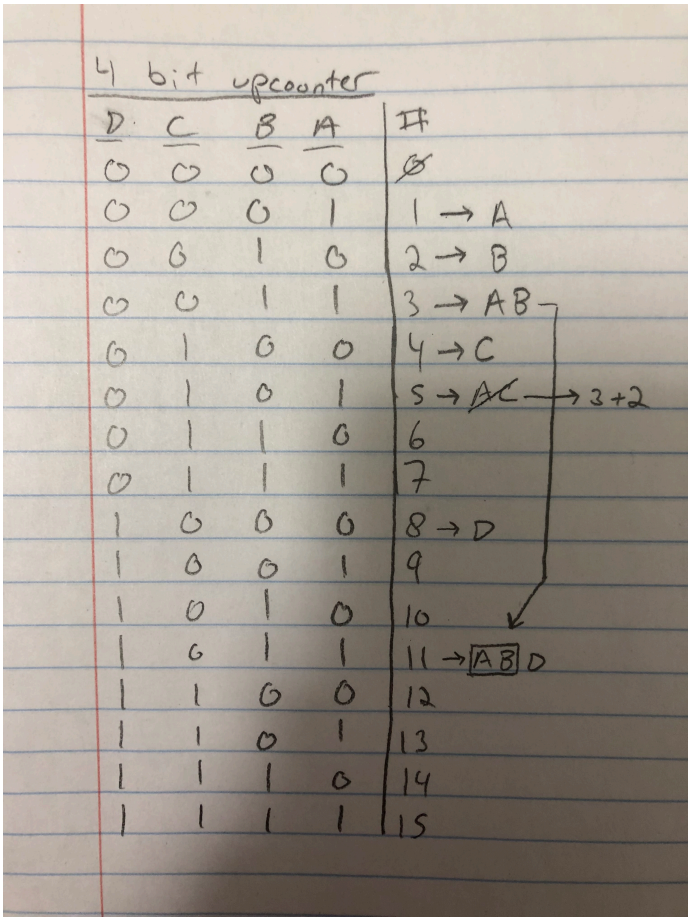


The state counter is used to select the input line of the mux. The timing system works by having logic connected to each mux input that will go high when it reaches the given number of seconds for the state on the corresponding line of the demux. For example, the first state is two seconds, so the mux input for the first input line will go high when the counter reaches two. The output of the mux are connected, since the inhibit pin is being used to differentiate, and sent to the preset pin of the timing counter and clock pin of the state counter. When the given time is reach and the mux output is a '1', the timing counter is reset to zero, using the jam and preset pins, and the state counter is clocked to the next state.

Since a four-bit counter is being used, I needed to separate the larger numbers. I decided that I should separate the larger numbers into the numbers that require the least amount of logic. First I found the logic for each of the necessary numbers. Then I derived the logic for every possible combination of numbers for nineteen and fifteen that would have at least one common number. I discover that eleven and eight/four only required one extra gate, while every other combination required two or three. I then realized that since I have an unused input line on the mux, I could split five into three and two to save a logic gate. This process can be seen below:





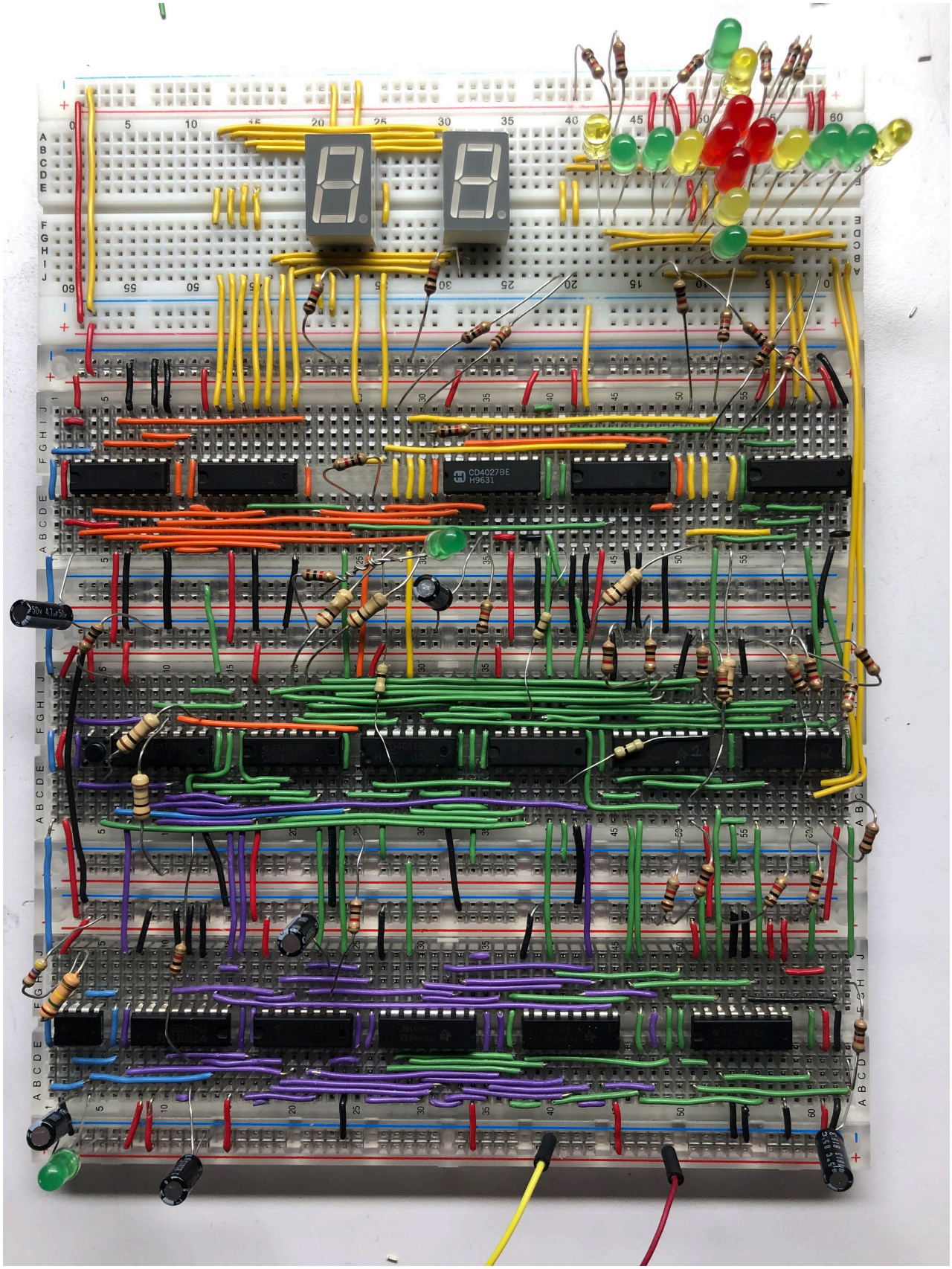


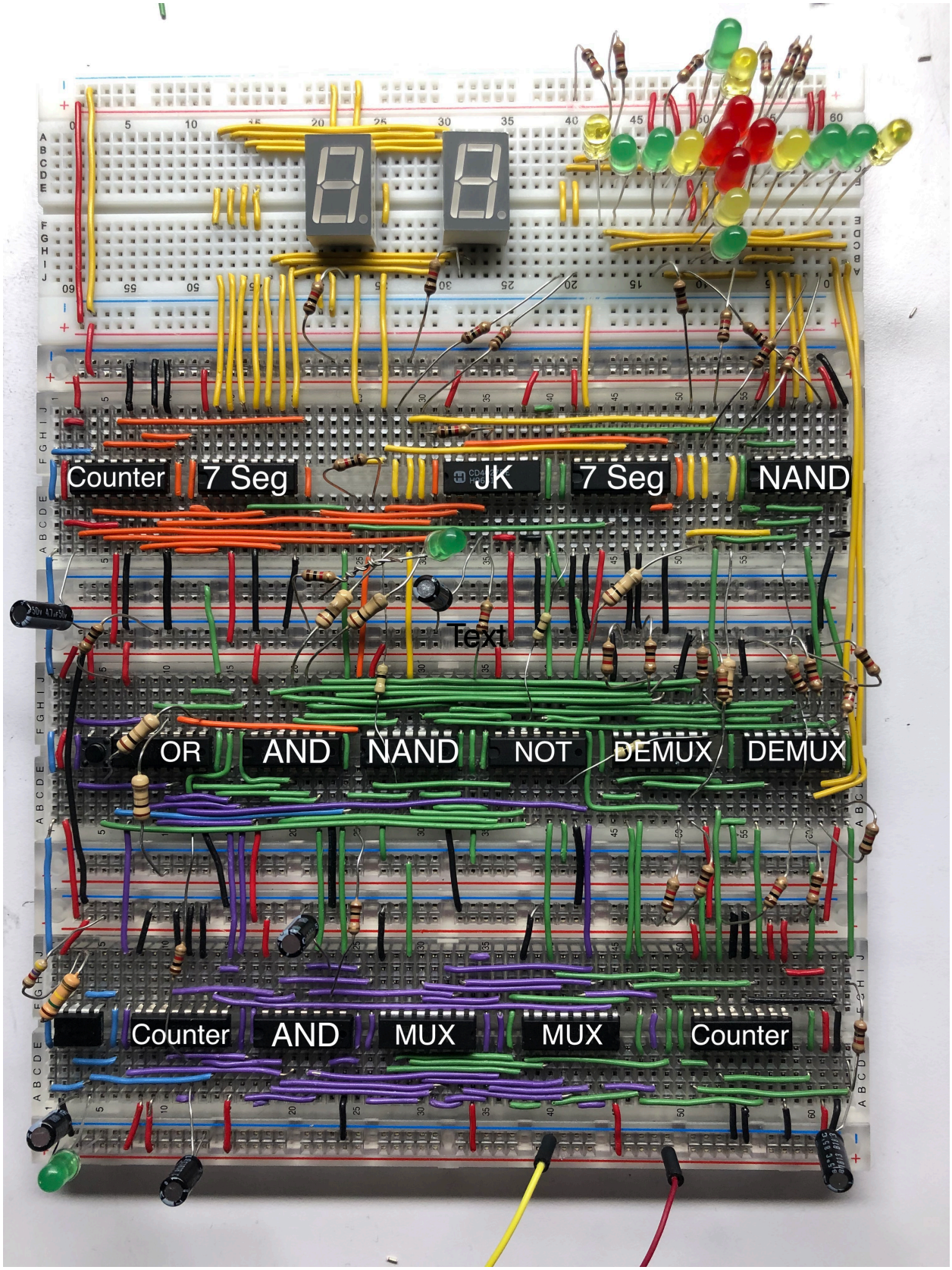
The next step of this circuit was the seven-segment display. Since my timing counter counts up, I required another counter that would count down for the seven-segment display. In order to correctly display the time remaining, I used two JK type flip-flops. Since the two green times have a ones digit of a nine or five, I found the common bits and attached the A jam pin to power and the B jam pin to ground. I realized that the D jam pin and the C jam pin toggled, so I used a flip-flop to accomplish that. To make sure that the flip-flop stays accurate, I have the reset pin attached to a NAND gate that outputs a '1' when D is 1 (on second demux) or when the button is pushed. This sets the jam pins to five. The clock pin is then attached to D' so it will toggle when it switched to the first demux. For the tens digit of the seven-segment display I used another JK flip-flop. I attached the D pin of the counter to the clock pin of the flip-flop and A' of the state counter to the set pin of the flip-flop, so when the counter reaches nine and the second

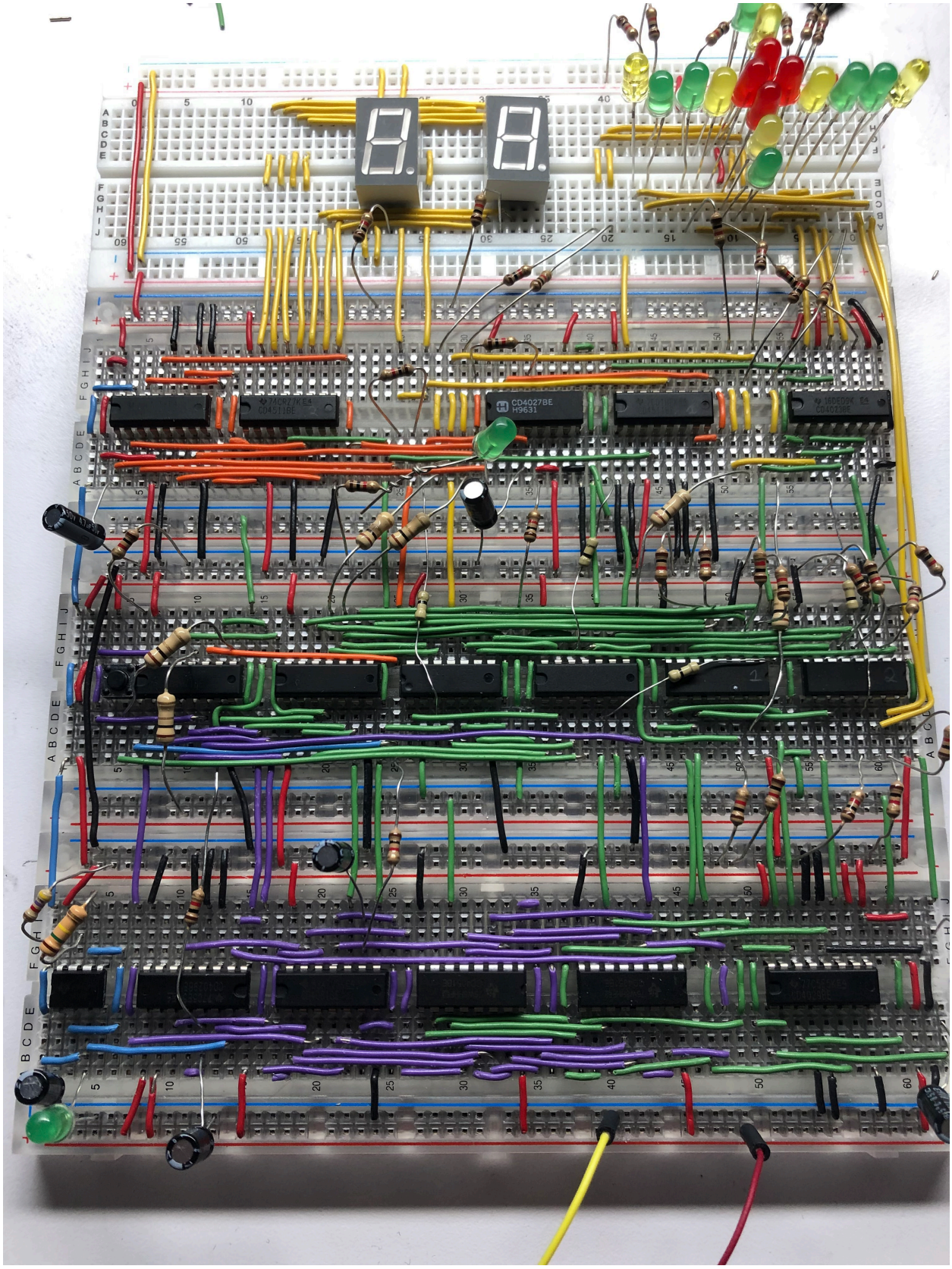
state of green is active the flip-flop will toggle to output zero. Additionally, the reset pin is attached to the button to ensure accuracy when the button is pressed. Similarly to the previous design, the button will only become active when the green LEDs of one direction are active and when there are more than five seconds left. When the button is pressed, the jam pins of the time counter are set to six, five seconds before it finishes, and the state counter is incremented if on the first part of a state.

Circuit Build

Below are pictures of the final circuit:


CD4027BE
H9631


Counter
7 Seg
JK
7 Seg
NAND
Text
OR
AND
NAND
NOT
DEMUX
DEMUX
Counter
AND
MUX
MUX
Counter


CD4027BE
H9631

Many problems arose while building. The first problem was lack of space. Although, the chip position and wires were planned, it was still a struggle to find space for each wire. There were also a few issues with timing. Since the logic chips are not ideal, there were slight delays that caused problems. This issue was solved using a resistor and capacitor to slightly delay the faster chip. Additionally, the same idea was used to de-bounce the button. Another issue that arose was inventory; the lab ran out of common cathode seven-segment displays, so my roommate and I had to share one for testing. Although this worked fine, one of the seven-segment displays we had was faulty – the left side does not light up [can be seen in on the tens digit in the video].

Conclusion

This project helped develop and hone many skills including wiring, design, boolean algebra, and working within the restrictions given. The project required precise, neat, and straight wiring, placed efficiently and logically, using a combination of creativity and critical thinking. Having to design two different possible solutions allowed me to hone my critical thinking skills and explore multiple avenues of design I would not have otherwise thought of. I decided on using the chosen design, because while it seemed to have more chips, the logic was less complex and seemed to require less wiring. Although the project works as it should, it could have been optimized further to require less chips and less wire. The complexity and size of the project gave me an opportunity to work systematically and efficiently on certain subsections that worked together to complete the whole. Although it was difficult to keep track of the wires, I believe building this project increased my skills and ultimately made me a better engineer.

Acknowledgements

I would like to thank Nathaniel Kingsbury for giving me the idea of using pull-up resistors for the outputs of the demux and giving me the idea to use resistors and capacitors.