BCD Adder

A project by Lucas Zapata

October 28, 2023

This is a project that adds two BCD numbers and returns the sum on a 7-segment display as well as a light. Components used include two SN74LS283N binary adders, one SN74LS00N NAND gate chip, a CD4511BE BCD to 7-segment chip, a dipswitch, and a 7-segment display.

Binary coded decimal (BCD for short) is a special form of binary that shows decimal numbers in a way easier for people to understand. BCD uses the same digit system as decimal, where the rightmost digit represents 10^0, the next digit represents 10^1, the next represents 10^2, and so on. Unlike decimal, each number in these places is represented by its binary equivalent. Table 1 shows how numbers are represented in decimal, binary, and BCD for comparison and further understanding of how these systems work compared to each other. Each digit in BCD must be less than or equal to 9. Adding BCD numbers is similar to adding binary or decimal numbers, however, its properties result in the implementation of new rules. Namely, if any of the digits in the final BCD number are greater than 9, 6 must be added to that digit. This is the equivalent of subtracting 10 due to the properties of adding two's complements. Figure 1 demonstrates a basic example of BCD addition. Now that the mechanic of BCD addition has been shown, a digital circuit that can accomplish this task can be built.

The circuit begins at a dipswitch, where the user can interact with the circuit. A user will input the two numbers to be added, which will then go into a 4-bit binary adder. From the adder, a logic circuit deducts if the sum is greater than 9. This circuit is the "error circuit". The building of the error circuit was done by first drawing a truth table which would return 1 upon the sum being greater than 9. Then, a K-map was solved that provided the most efficient circuit for this function. Then, by DeMorgan's Law of Boolean algebra, the circuit was simplified to use only the NAND function. This process is demonstrated in Figure 2.

The previously acquired 4-bit sum is sent into another binary adder, where it will be added with 6 or 0 based on the output of the error function. This was done by wiring the output of the error circuit into the second and third bit of the adder.

The output of the Error circuit is also sent to an LED which represents if the sum of the two inputted BCD numbers is greater than 9. The LED essentially represents the tens place of

the sum. Because 4-bit BCD numbers can only represent numbers from 0 to 9, the maximum sum of two 4-bit BCD numbers is 18. Because the tens place can only be either a 0 or 1, an LED is sufficient to represent it.

After the second binary adder, the sum is sent into a BCD-to-7-segment display integrated circuit, which is finally hooked up to a 7-segment display. The display represents the ones place of the sum, and the aforementioned LED represents the tens place of the sum. Now, we have a fully functioning BCD adder which presents its sum to the user in decimal format. Figure 3 shows the final circuit diagram and Figures 4-8 demonstrate the circuit working.

Building the circuit, however, was not as simple as just putting wires together. One of the first problems I encountered early in building was the quality of my wires. Initially I used pre-cut wires, which gave me many problems. They were difficult to maneuver around the chips as well as being very difficult to decipher where they were going. I then switched to jumper wires because they provide more flexibility and allow me to follow their paths more easily.

Another problem I ran into early in the building was something going wrong in the error circuit. The error circuit would result in the LED being on when it was not supposed to be and off when it should not have been. There were few sums where it gave the reading it was supposed to. To begin troubleshooting this error, I began by examining if something was wrong with my dipswitch. I wired every output of the dipswitch to an Arduino board. I wrote a small software that would read what my dipswitch output and formatted it into two BCD numbers. The error was not in the dipswitch, so I moved on to troubleshooting the first binary adder. Similar to how I troubleshooted the dipswitch, I wired the outputs of the adder to the input pins of the Arduino and wrote a script that would show me what the adder was saying. To my surprise, the adder was not outputting what it should have. I quickly inspected the wiring and found that the error was there. The circuit functioned as normal from there.

The next problem once more had something to do with the error function. I realized that the adder would work near perfectly (the final issue will be discussed later) with the exception of numbers greater than 15. Initially, the error circuit did not account for the carry-out of the first adder, as shown in Figure 9. 4-bit binary can represent numbers from 0-15 but needs a fifth bit to represent numbers 16-31. This is what the carry-out on the adder is for, to output that a fifth bit exists from the addition, even if both addends have only four bits. This was a problem and made the circuit unable to identify that the numbers 16-18 were greater than 9. The initial error circuit would only add 6 if it read 11XX or 1X1X (where X means that the bit can be either 0 or

1). However, in the cases of numbers 16-18, they would present as 10XXX, which would not trigger the error function. I solved this problem by adding the carry-out to the NAND of the error function, which now allowed it to properly identify the 10XXX case.

The fourth and final problem is the only one that still exists within my circuit. The circuit works for all possible BCD addends with the exception of two sums. As shown in Figure 10, the circuit completely turns off when the sum is equal to 12 or 13. Neither the LED nor the 7-segment display return anything. I am unsure what causes this problem, but further analysis could be done by running power through the LED while it is in this state or using an Arduino to analyze the output of the CD4511BE chip.

With all said and done, there are a few changes I would like to make in a more finalized version of the project. First, I would attempt to find a solution to the problem in the previous paragraph more thoroughly. Second, I would use a higher quality 7-segment display, since the one I am using is quite finicky and has some LEDs turn off when it is wiggled around. I would also use higher-quality resistors that fit better in my breadboard, as those were another issue I commonly ran into. The result would not work, but upon wiggling around some wires and resistors, the right result would display. In fact, if finalized, it would be nice to design and order a custom PCB that I could solder connections onto and make a more stable, professional product.

This project has been a very valuable experience for me. This was my first time working on a relatively complicated circuit with many wires and different chips to account for. I learned how I can troubleshoot a project thoroughly with the use of an Arduino, and some common problems I may face through my electronics journey. I am very proud of my work and am excited for future projects I will undertake, all hopefully much larger and more complicated in scale than this one.

Appendix

Table 1: Decimal, Binary, and BCD comparison

Decimal	Binary	BCD
7	0111	0111
24	11000	0010 0100
313	100111001	0011 0001 0011

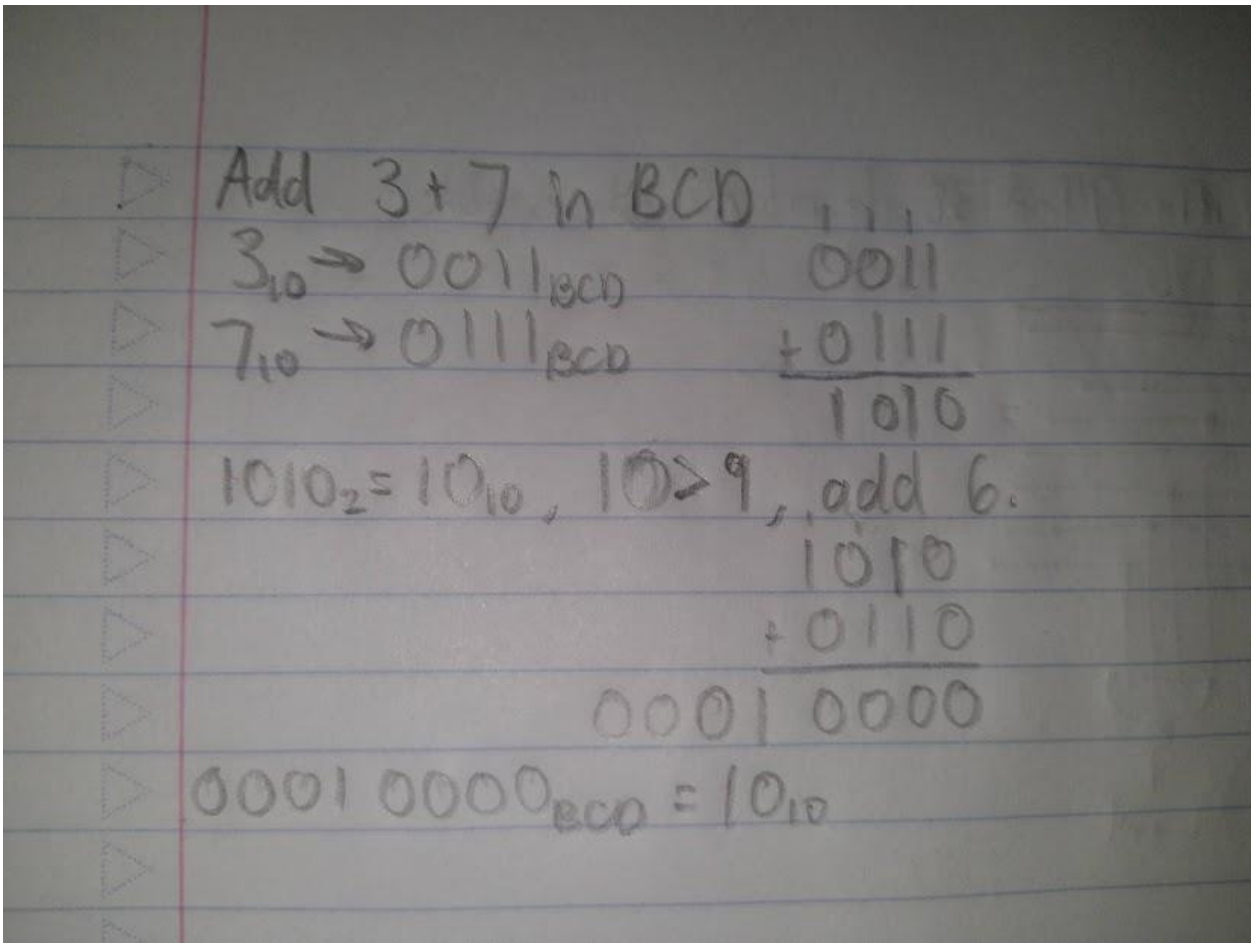

Figure 1: Example of BCD Addition

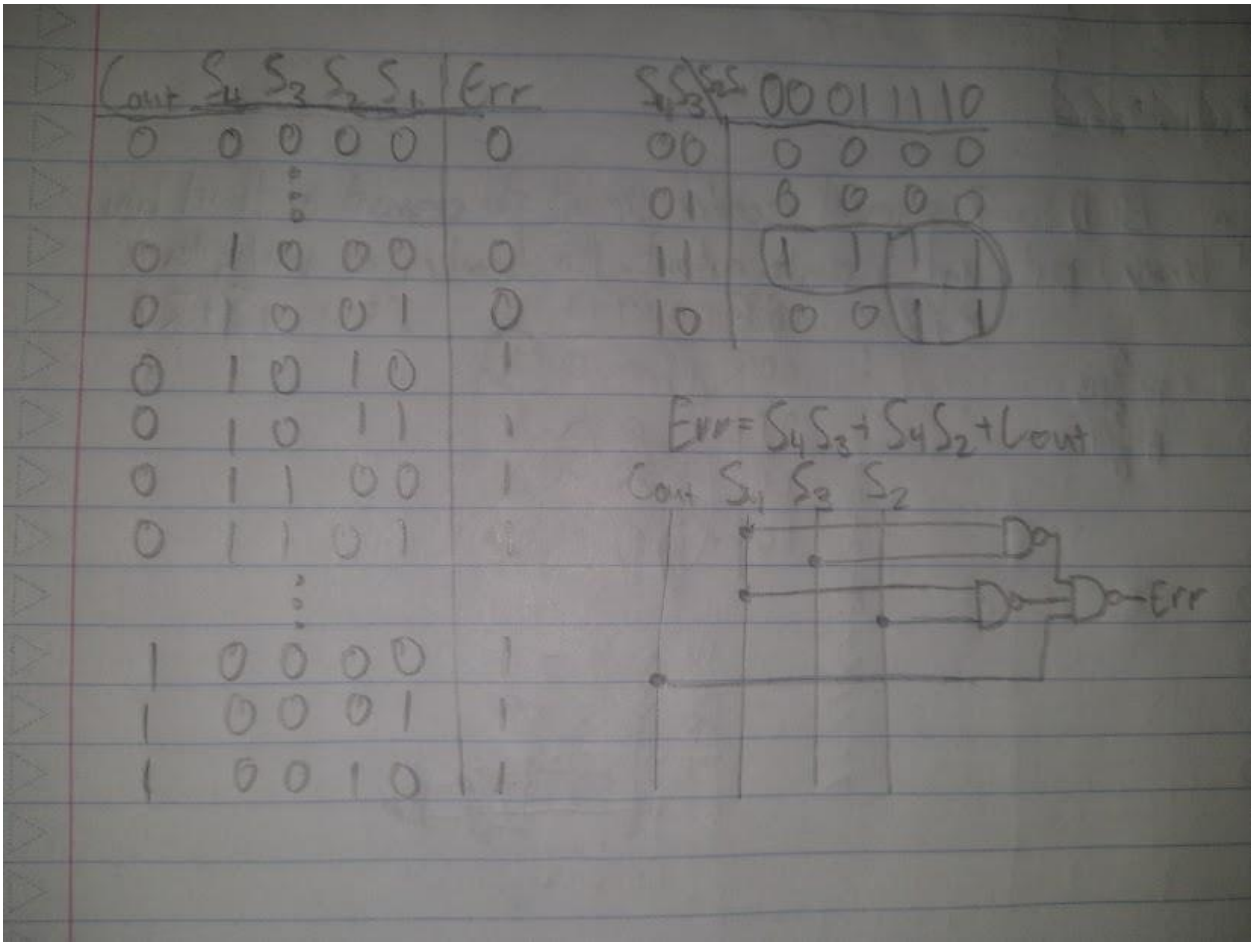

Figure 2: Acquisition of the Error Circuit

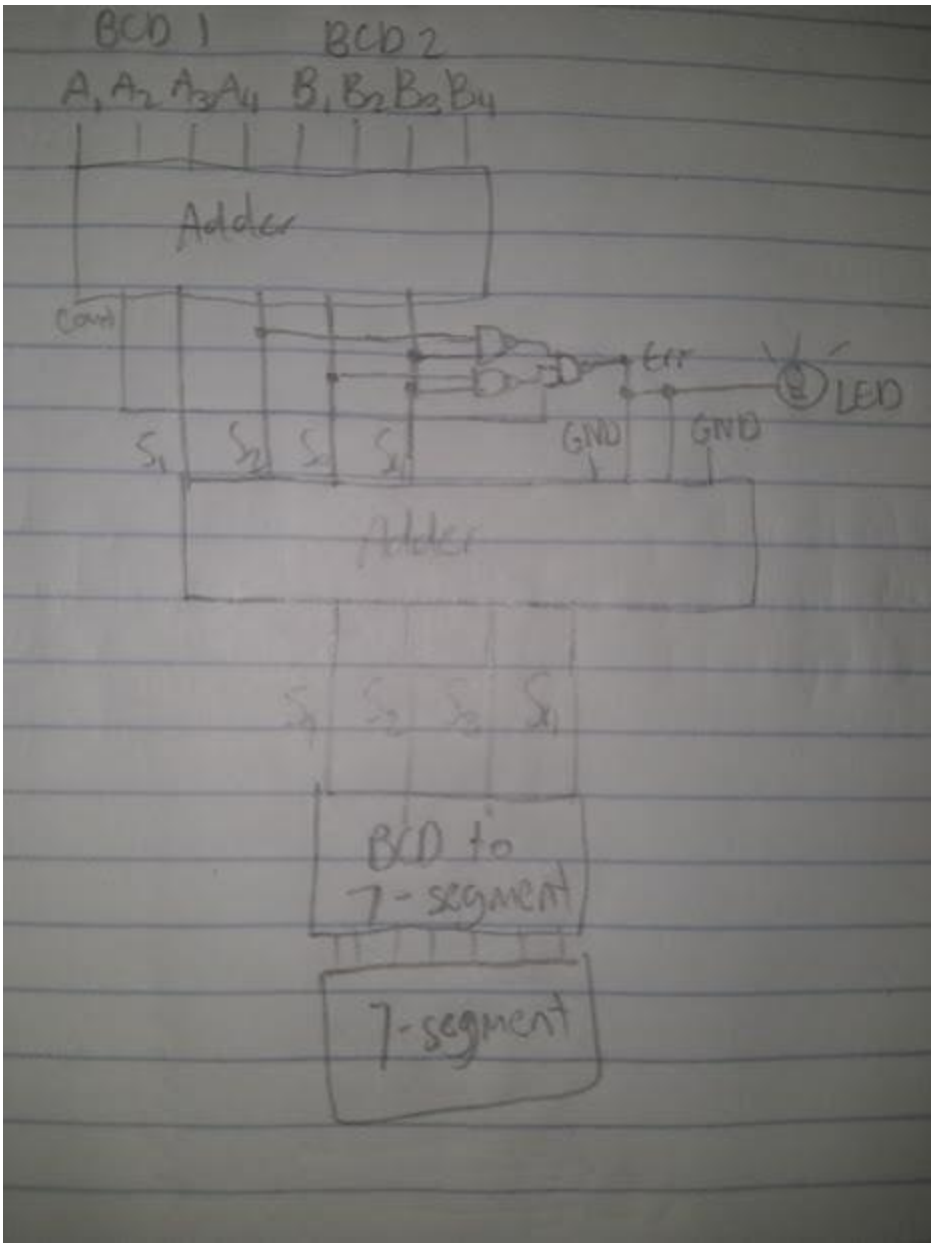

Figure 3: Simplified Circuit Diagram

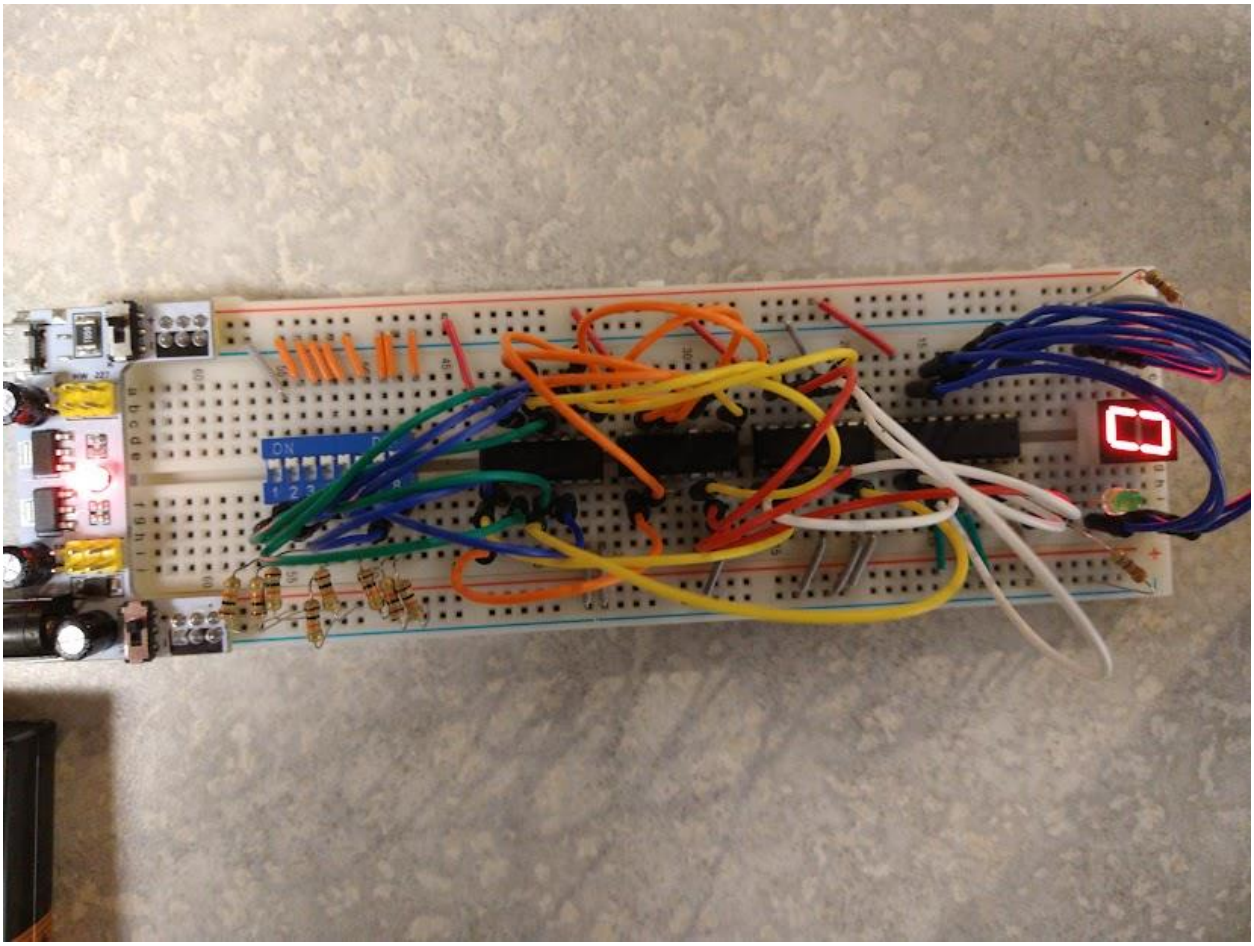

Figure 4: 0 + 0 displays 0

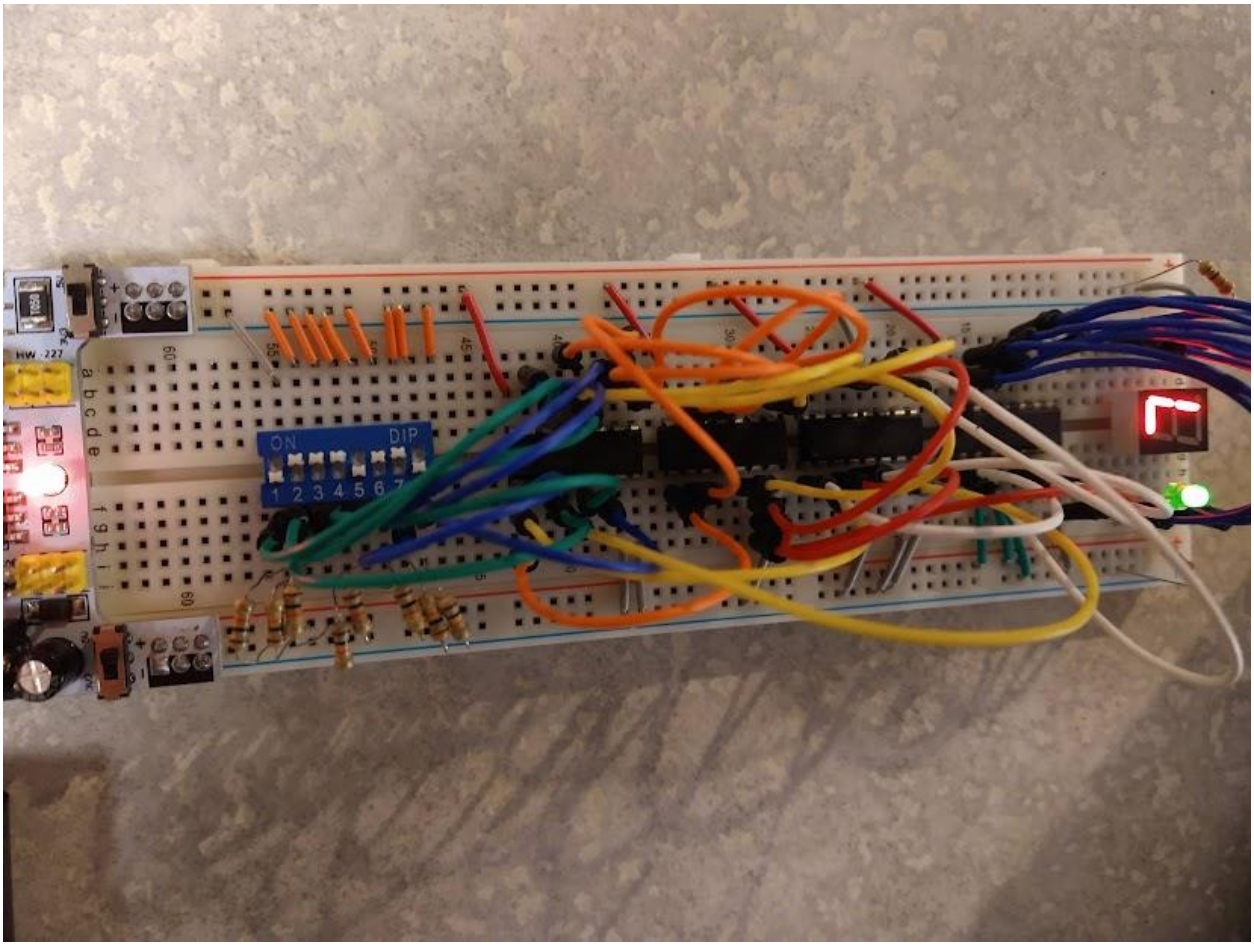

Figure 5: 8 + 9 displays 17

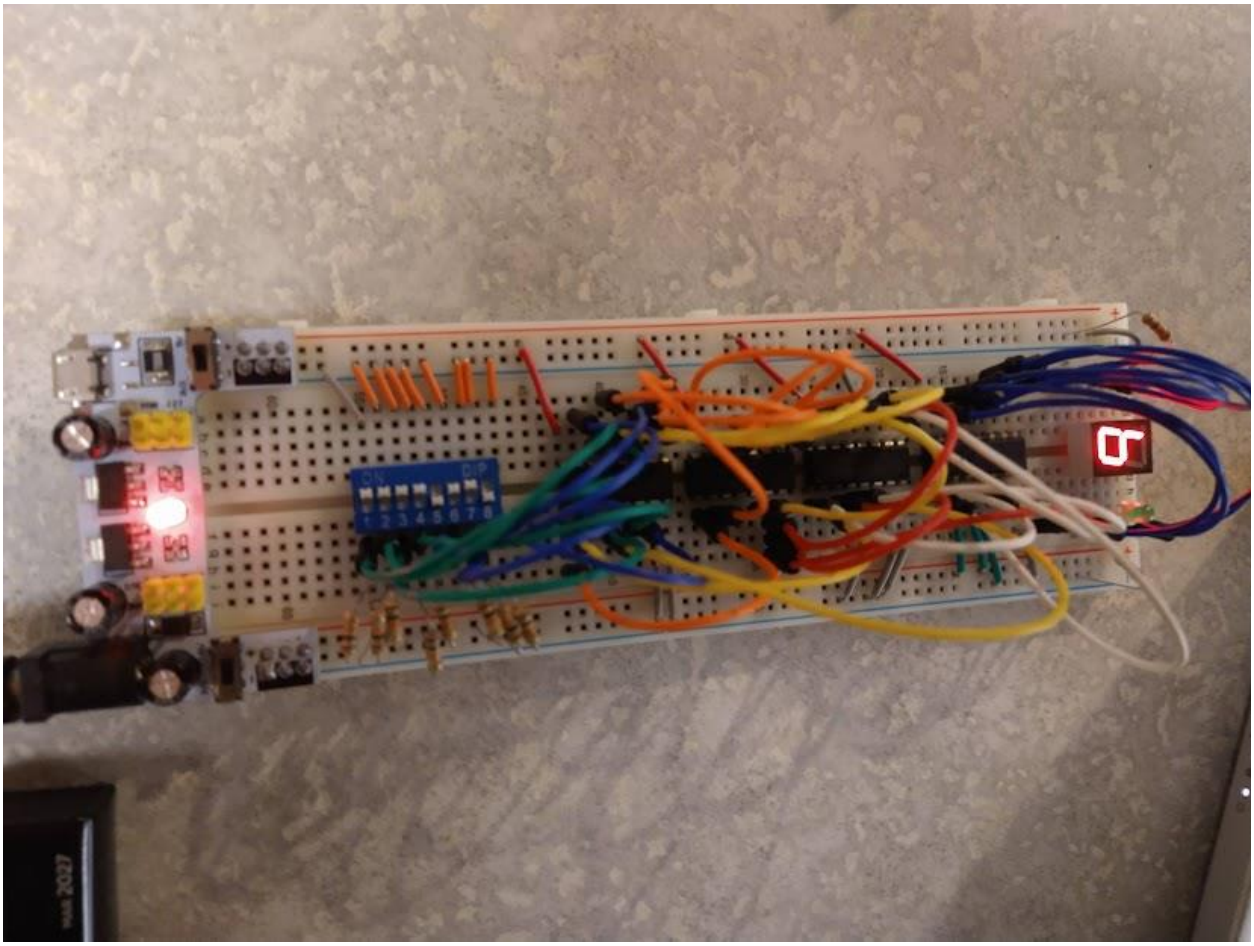

Figure 6: 0 + 9 displays 9

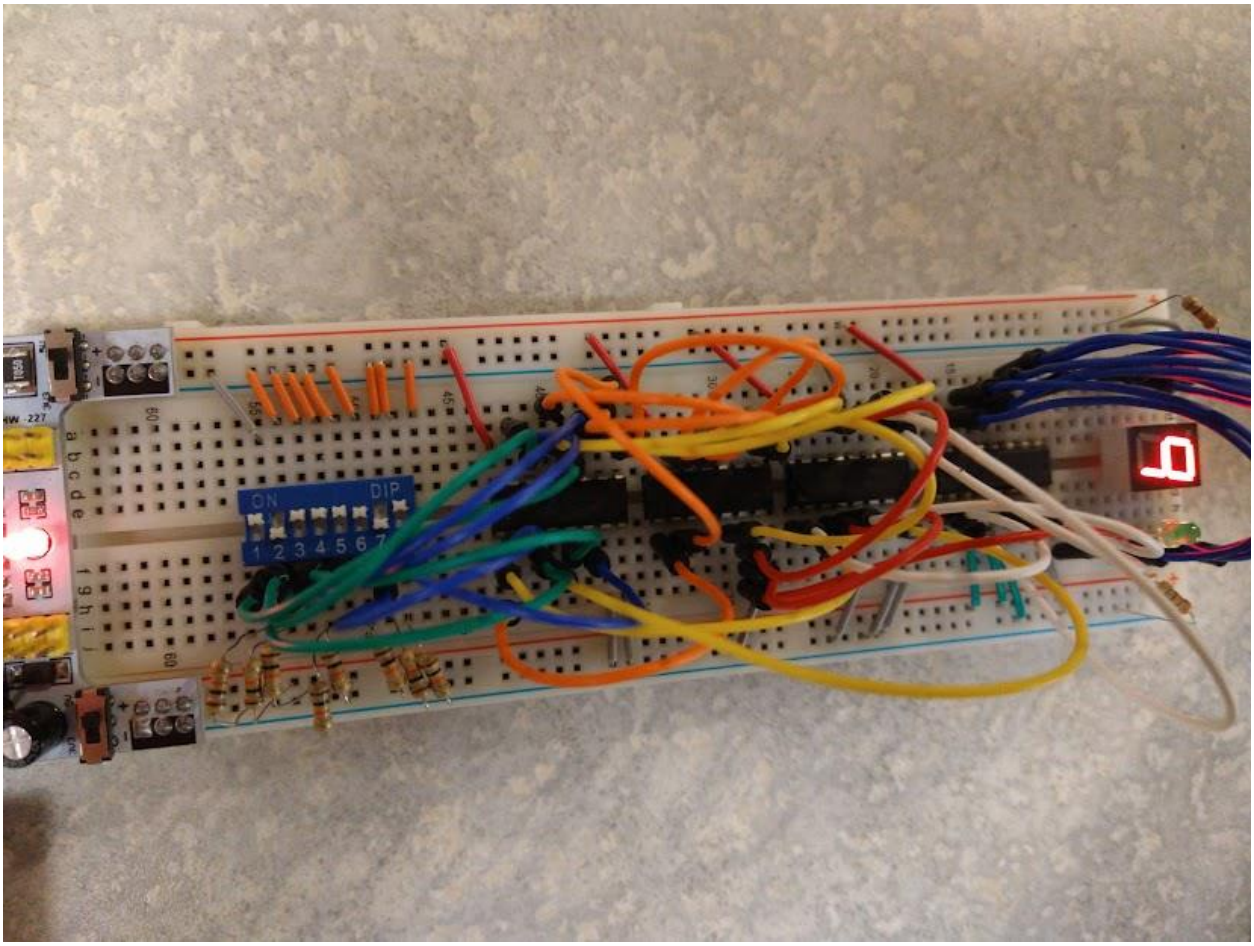

Figure 7: 4 + 2 displays 6

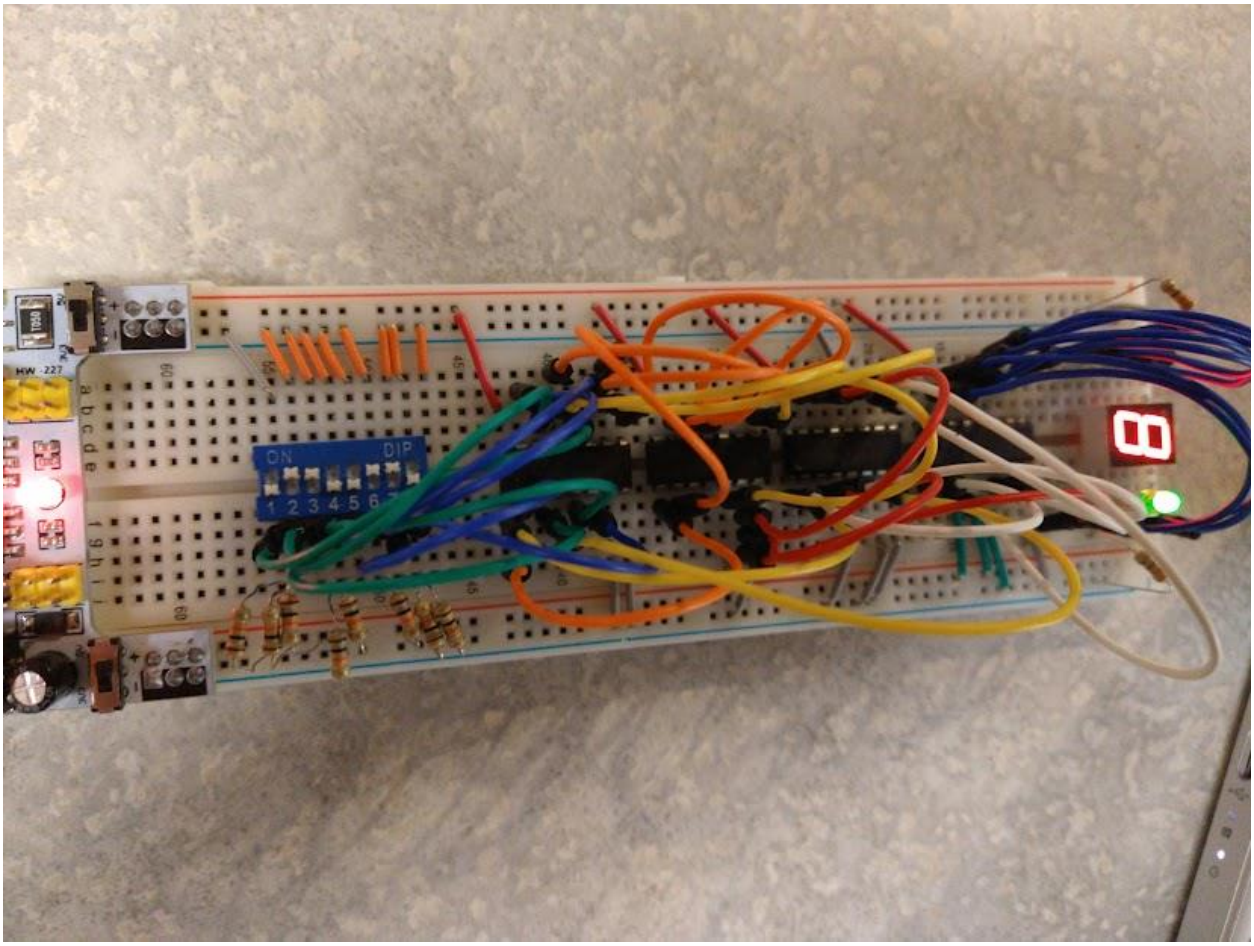

Figure 8: 9 + 9 displays 18

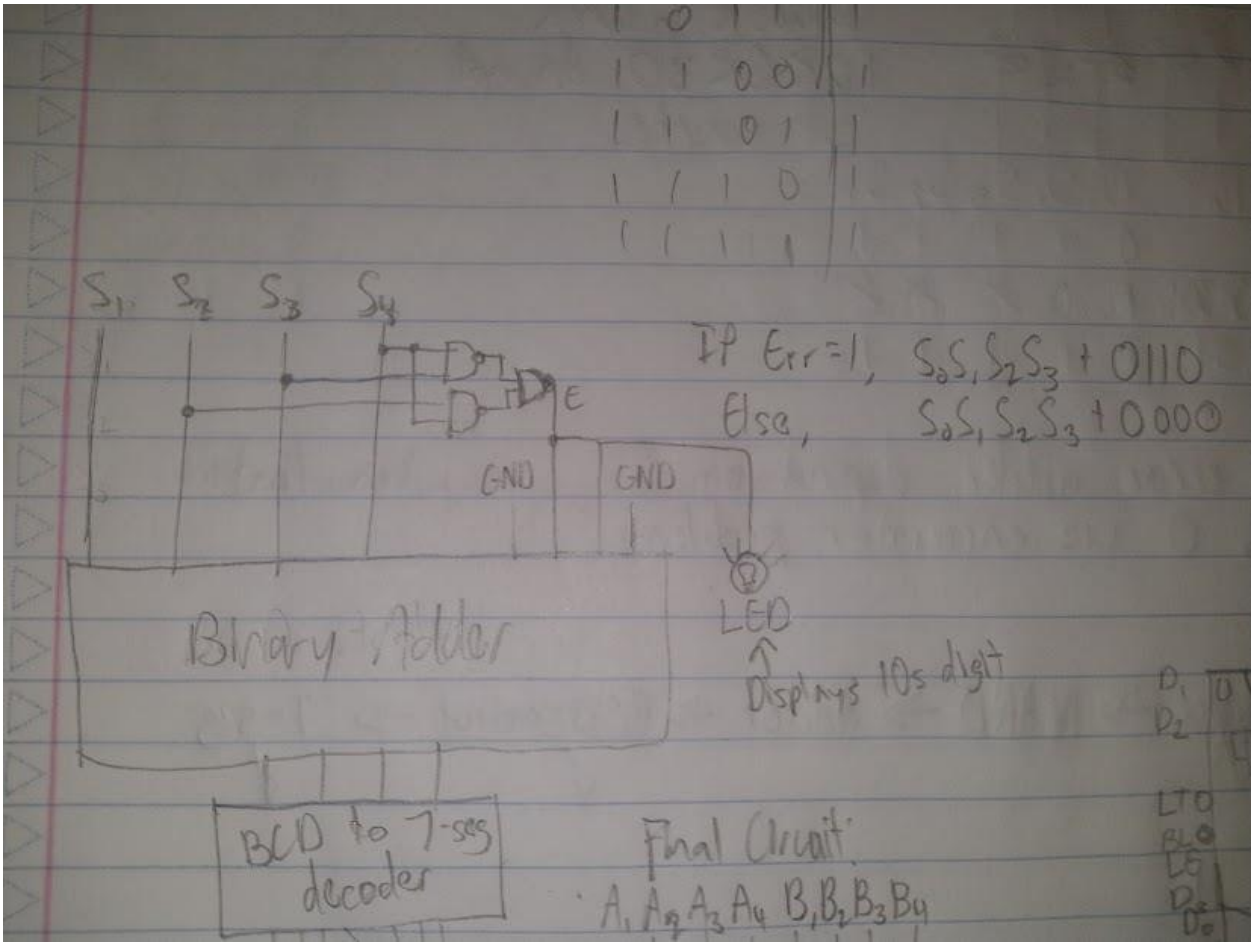

Figure 9: First Rendition of Error Circuit

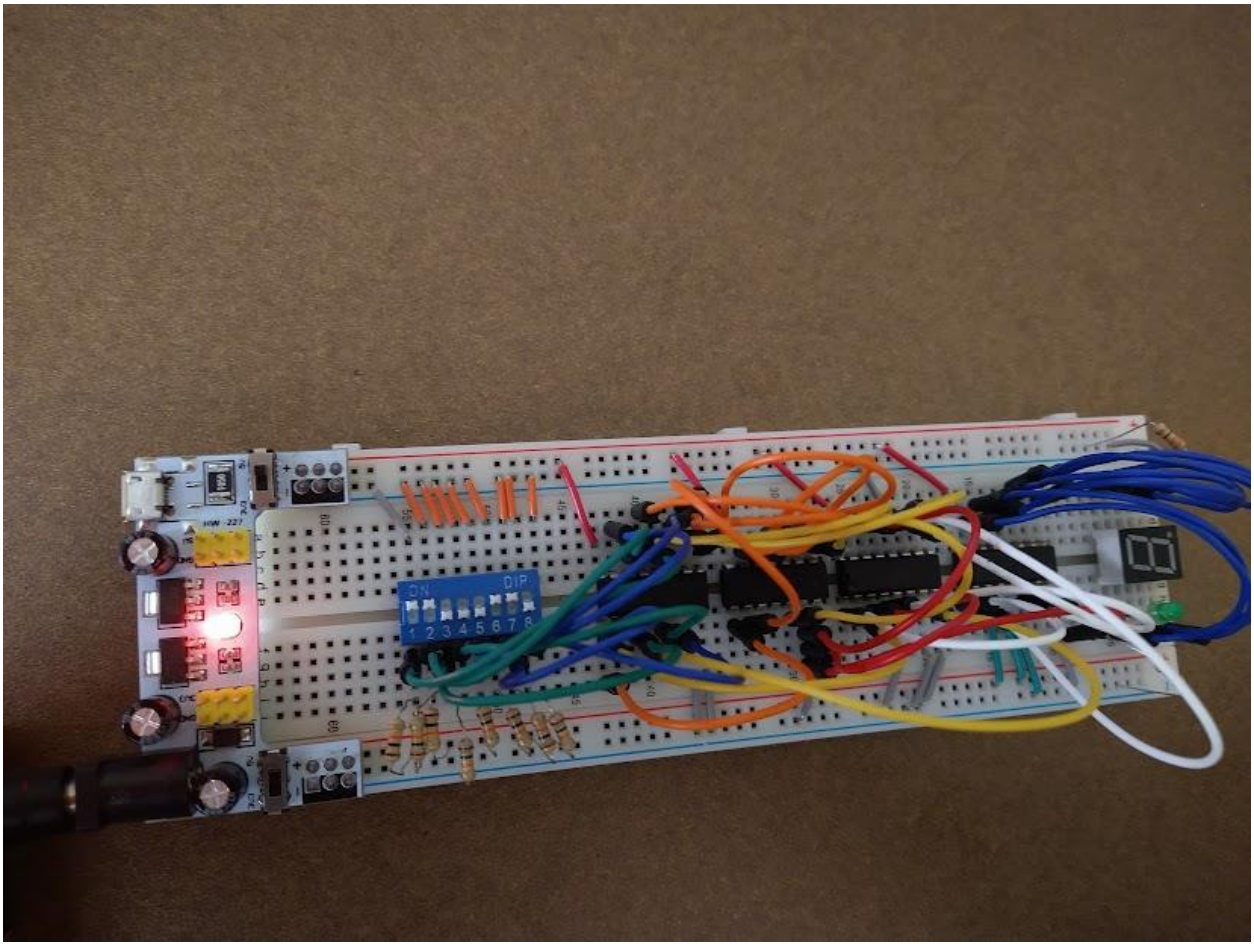

Figure 10: 3 + 9 turns LED and 7-segment display off entirely