8-Bit Adder-Subtractor Circuit

Department of Computer Science

Spencer DeMera

spencer.demera@csu.fullerton.edu

CPSC 440: Computer System Architecture

Semester Project Report

27 November 2022

1. Project Description

I originally started designing this project in August in anticipation of taking this class as well as a way to explore my interests. I am hoping to one day go on to pursue a more computer hardware focussed career and potentially a hardware/embedded systems focused master's degree.

The project that I decided to build for this semester is an 8-bit adder-subtractor which can also be used as an extremely simple Arithmetic Logic unit. I developed two separate versions of this project via circuit design software, both of which can function extremely well and fullfill their purpose. I additionally developed the simplified second circuit in hardware with breadboards once my virtual designs were complete. It is merely a proof of concept and exercise of my knowledge. This physical component is built entirely in hardware with 7400 series TTL chips, breadboards, and wires. This circuit works entirely in 8-bit binary via the TTL chips and I/O modules. Input is taken in via 8-switch dip switches and output is displayed across individual LEDs in binary representation. Addition will be displayed in normal signed magnitude and subtraction will be displayed in 2's complement binary form. There are also three flag LEDs for the purpose of declaring a leftover carry out/integer overflow, a zero flag, and a sign flag (for the purpose of declaring whether an output is negative). Additionally, to control whether or not to do addition or subtraction there is another dip switch that acts as a control input. High control input denotes subtraction while low control input denotes addition, additional is default.

2. Project Design

2.1. Software Design

I did all of my designing and architecting entirely in software first for learning purposes and also to ensure I could easily copy and redo components. Primarily for basic logic design I used Logisim Evolution as my tool for the entire schematic. I designed every component of this circuit with Logisim and even went as far as to develop a full ALU with bitwise operations as well, though it's far too complex to implement. After each component was designed separately and I felt especially confident, I went through and developed multiple versions of my 8-bit circuit. The more complicated of the two does addition, subtraction, and conversion from 2's complement to signed magnitude. The simpler of the two is vastly slimmed down and optimized and does no such 2's complement conversion, while also making a much more efficient use of logic gates.

The simplified design that I used as my final schematic, at an abstracted level, is based off of the simple 8-bit Adder-Subtractor circuit from *Digital Computer Electronics* by Albert Paul Malvino, Ph.D. Following this design allowed me to remove the unnecessary extra logic gates and simply output the result value in 2's complement binary form. Once a design was finalized, I moved onto hardware research and eventually built a virtual breadboard schematic of the I/O module and 8-bit adder portions of the design in Tinkercad. Figures presented below:

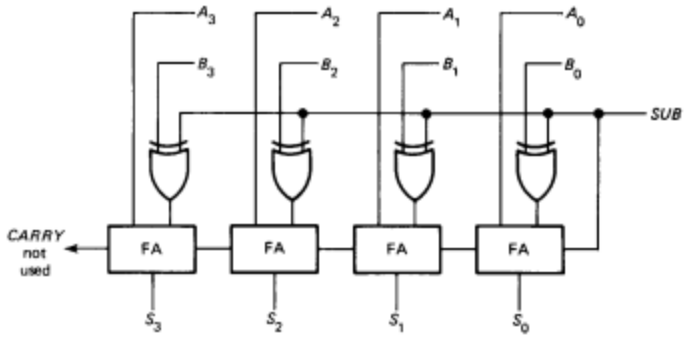

Fig. 6-7 A 2's-complement adder-subtracter.

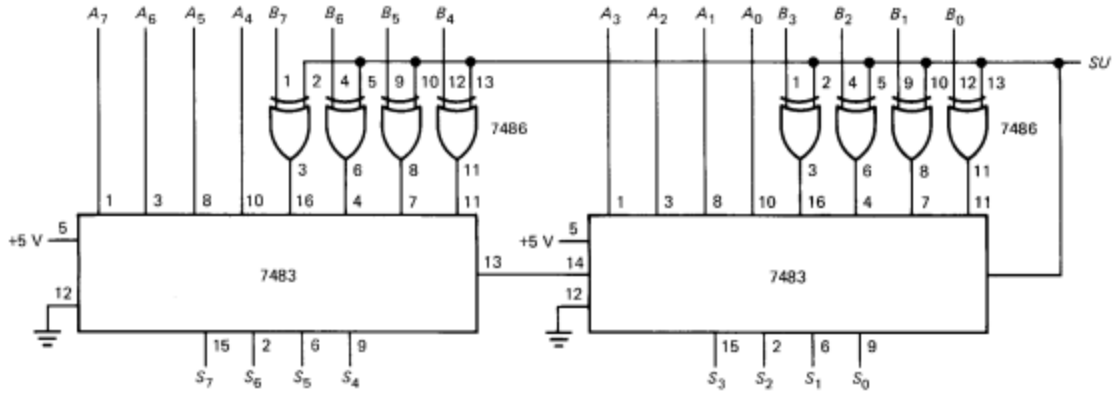

Fig. 6-8 TTL adder-subtracter.

Diagram 1.

TTL Adder-Subtractor abstracted design from *Digital Computer Electronics* by Albert Paul Malvino, Ph.D.

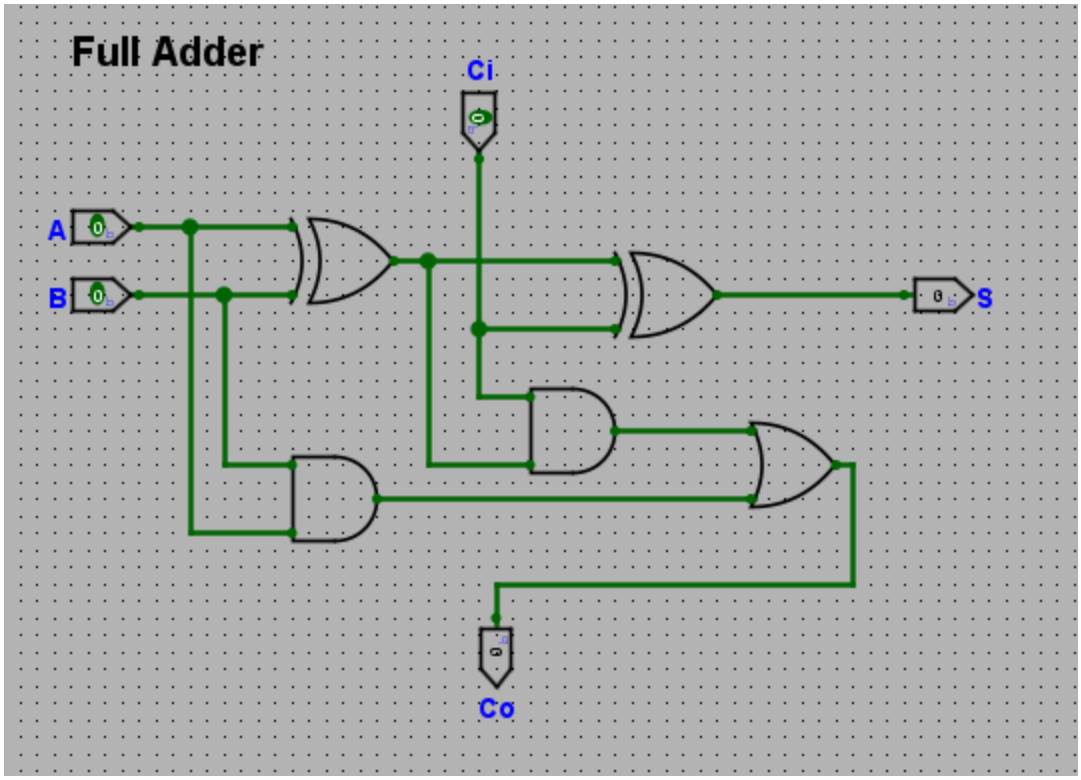

Diagram 2.

A full adder circuit designed in Logisim Evolution. This circuit consists of 2 Positive XOR gates, 2 Positive AND gates, and 1 Positive OR gate.

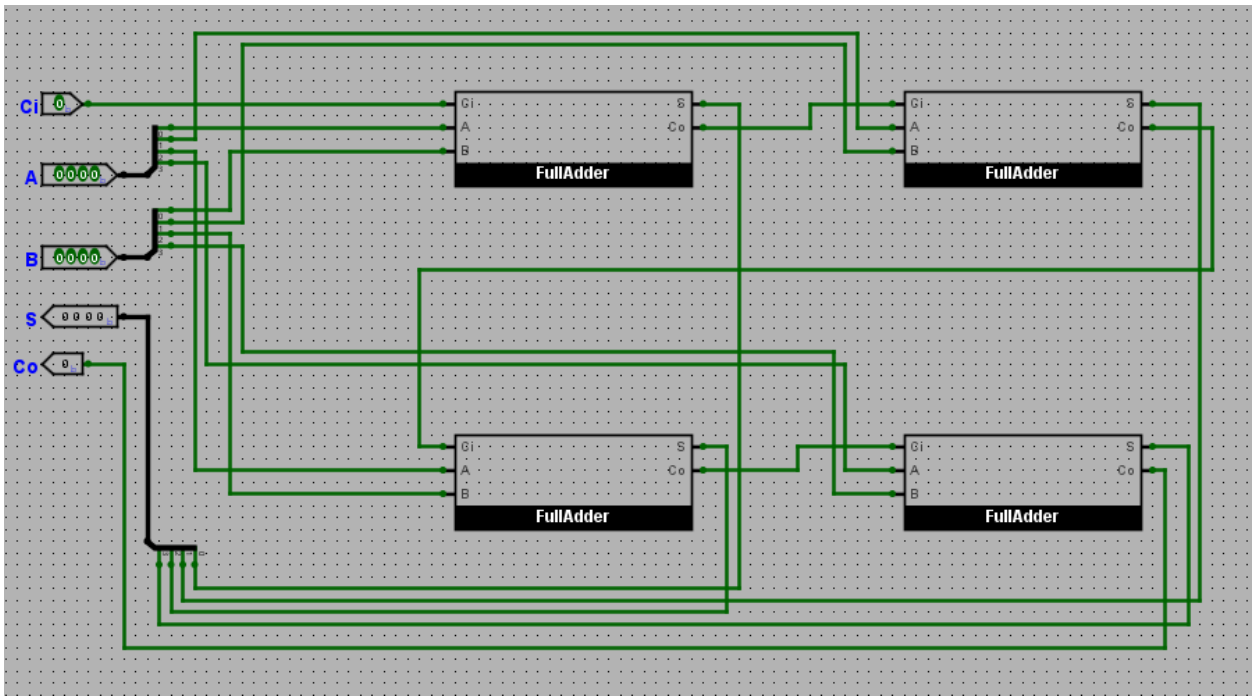

Diagram 3.

A 4-bit adder circuit designed with four of the full adder circuits from diagram 2. This circuit was a proof of concept for linking together multiple full adders for later use.

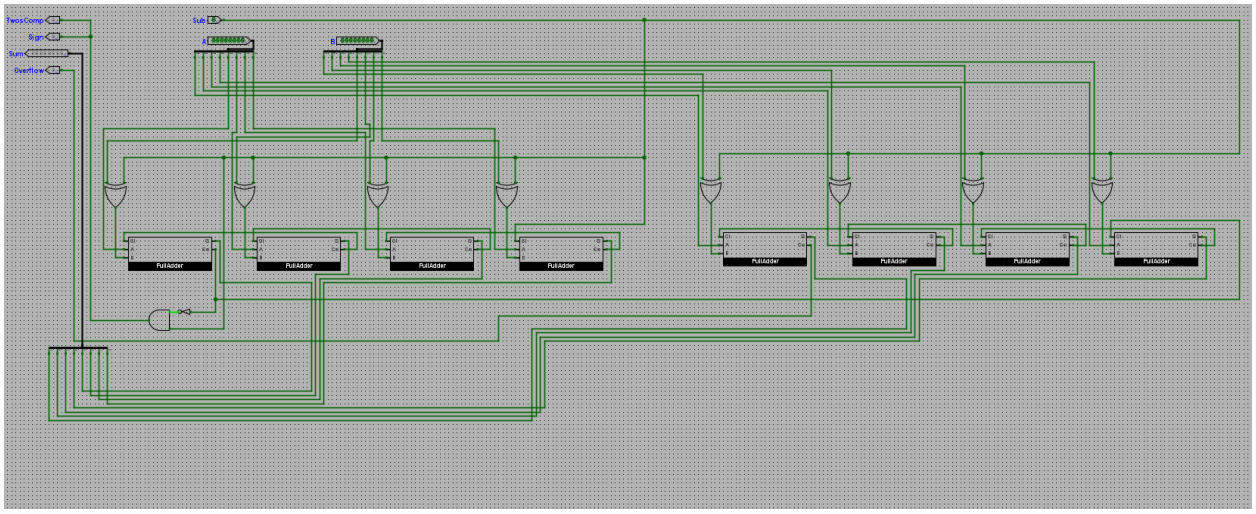

Diagram 4.1.

The completed schematic of the fully functional 8-bit adder-subtractor circuit. This circuit links together eight full adders in the same fashion as diagram 3. These adders are split into two different blocks and have their own corresponding sets/blocks of Positive XOR gates for managing carry input. This design is the one that is based off of the one in Diagram 1. As mentioned prior.

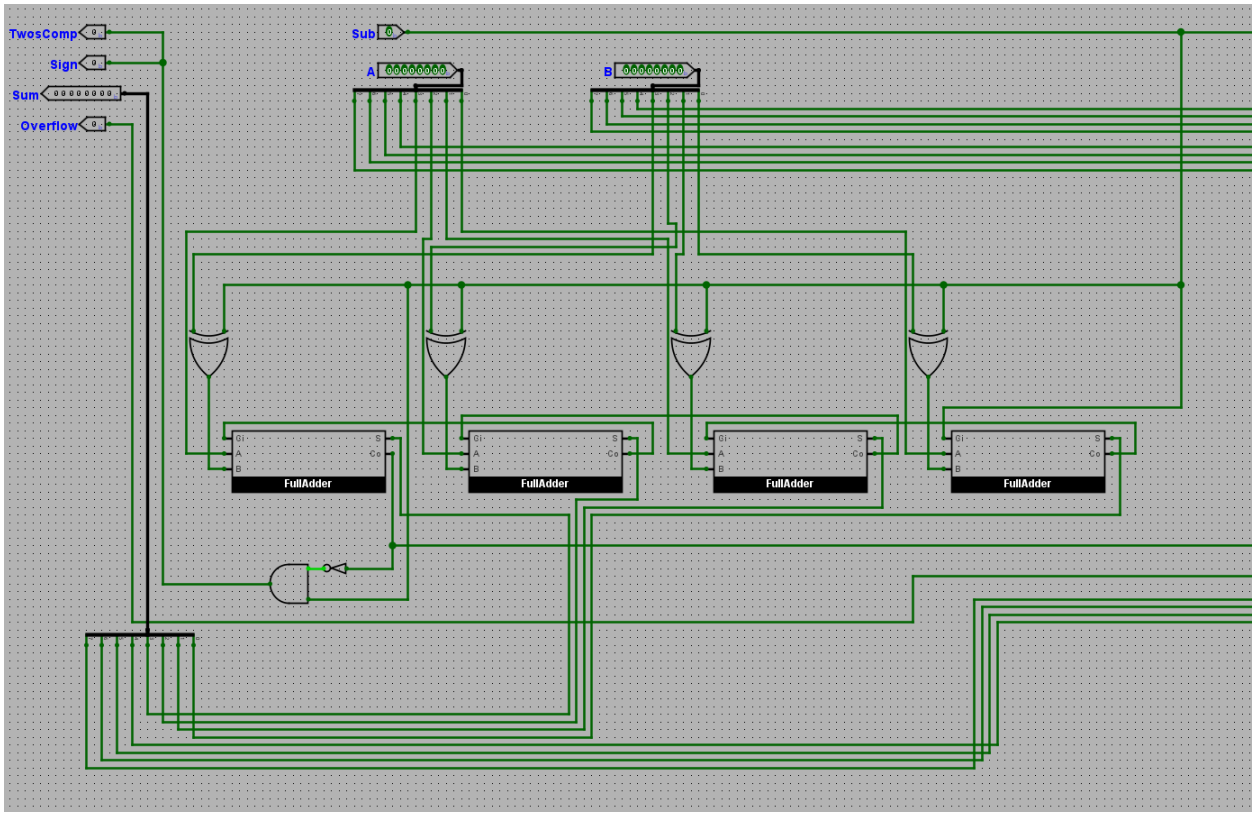

Diagram 4.2.

This is a closer look at the schematic architecture of the I/O systems as well as the general connection structure linking everything together. All wires leading off to the right of the image go on to the second block of logic.

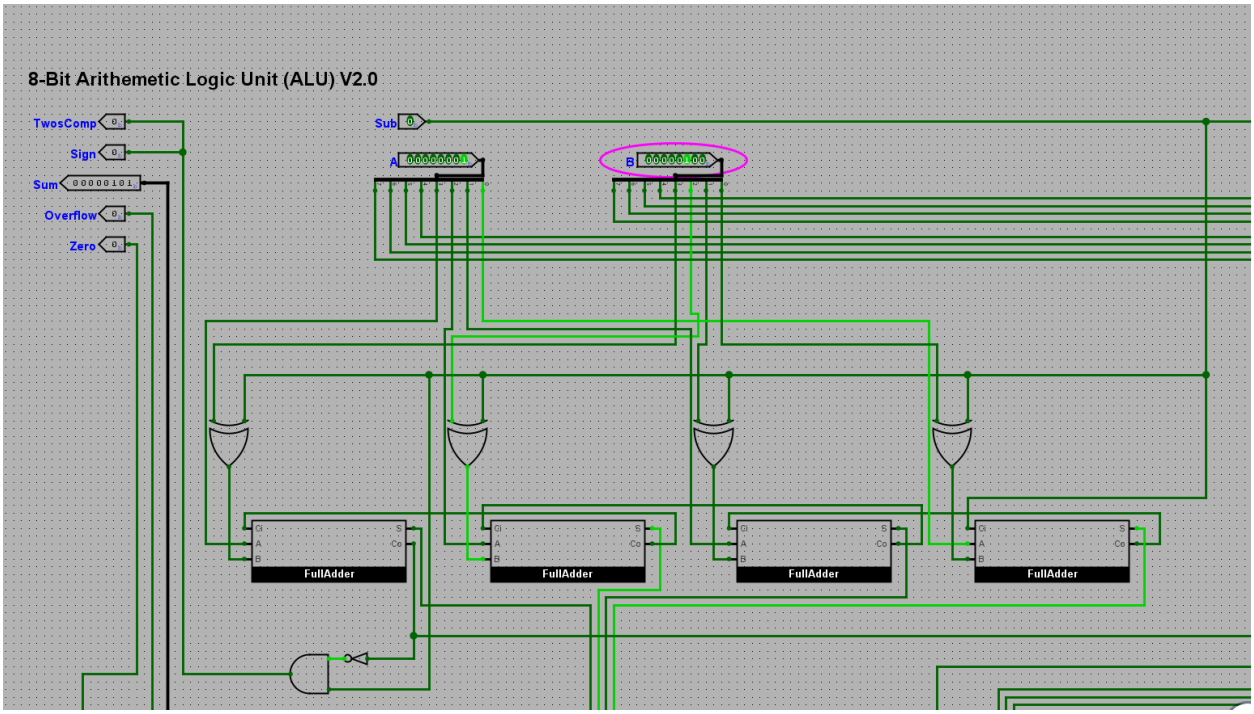

Diagram 4.3.

Here is the same circuit from 4.1. but computing in the addition state. Here it is computing `00000001 + 00000100 = 00000101`

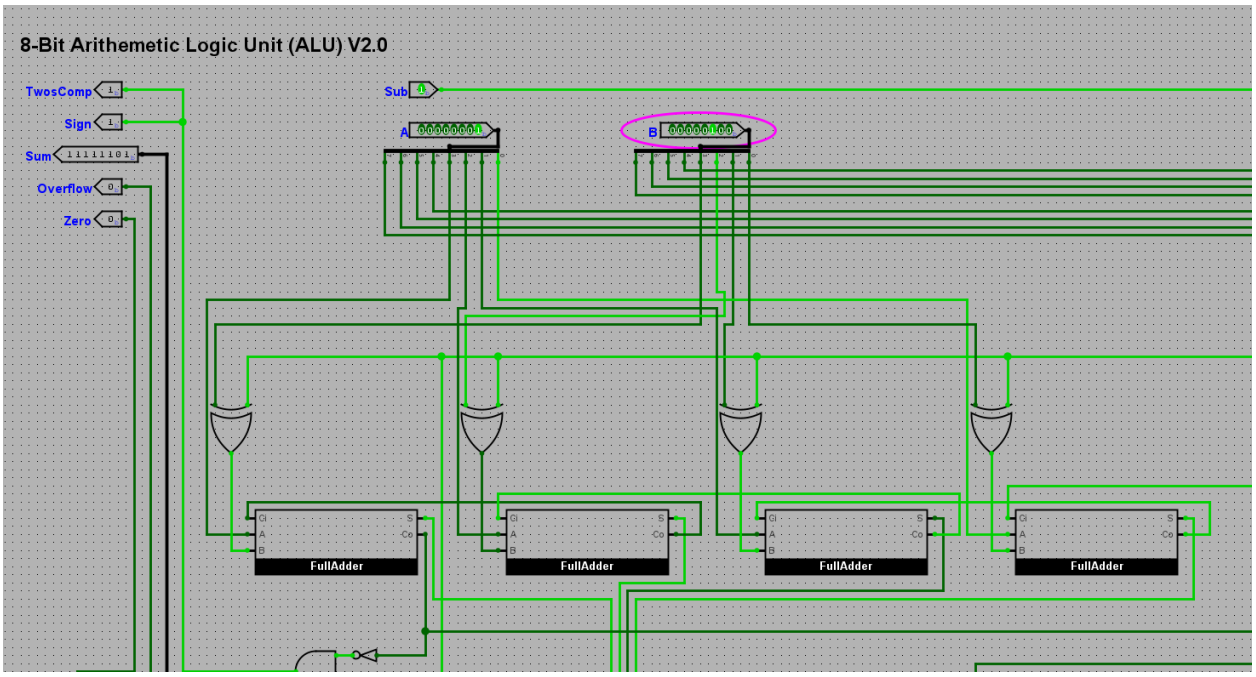

Diagram 4.4.

Here is the same circuit from 4.1. but computing in the subtraction state. Here it is computing `00000001 + 00000100 = 11111101`

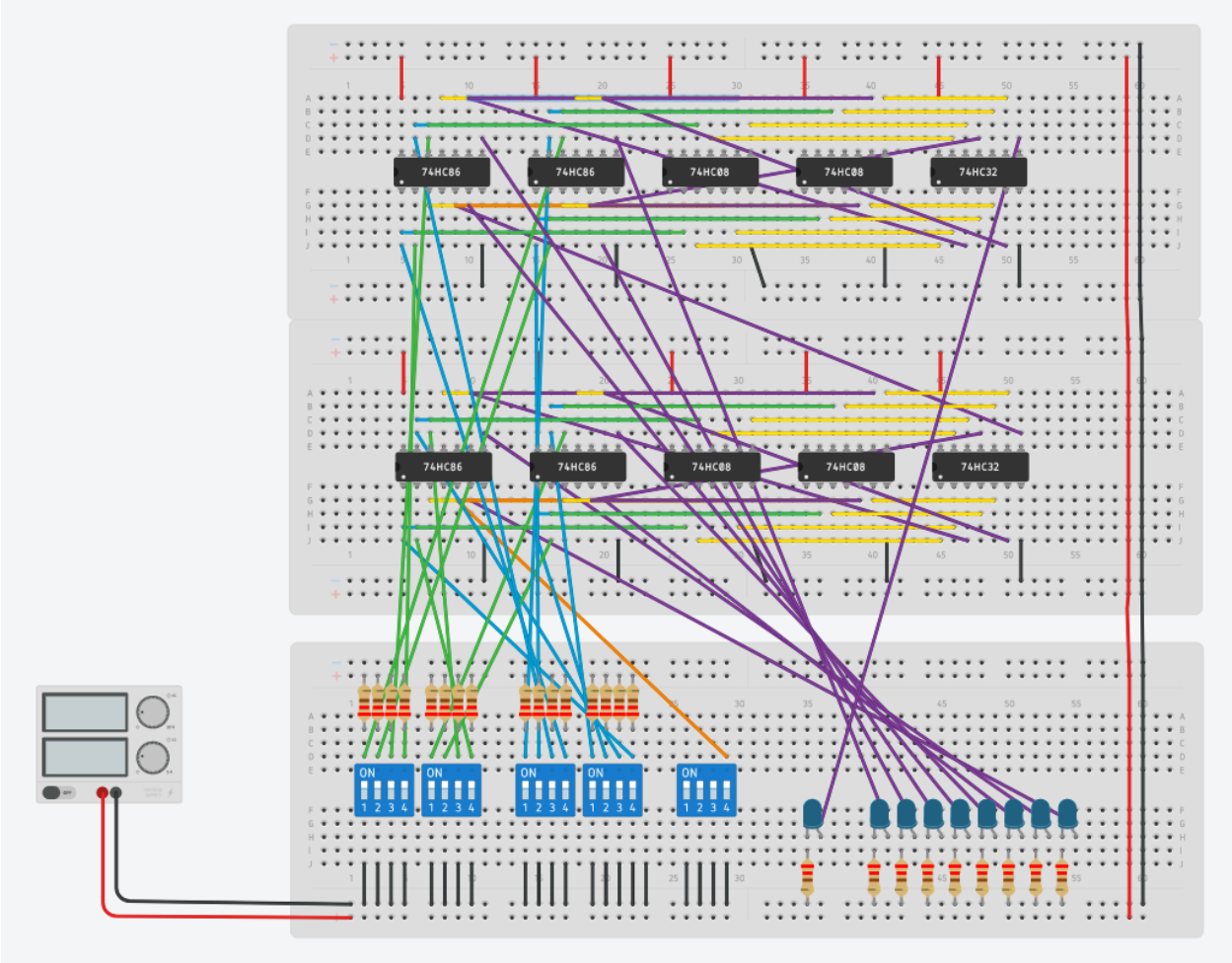

Diagram 5.

This is the virtual hardware design of the I/O module and 8-Bit adder used for the project. This circuit diagram was completed in Tinkercad and contains all relevant circuitry for an 8-bit adder but not the final 8-bit adder-subtractor. This schematic serves as a closer-to-physical hardware proof of concept design.

2.2. Hardware Design

Once designing was completed in software I began to build the circuit in actual hardware with breadboards, jumper cables, resistors, etc. This building occurred side by side with various virtual designs to ensure I had plenty of time to learn, assemble, and troubleshoot as I went. Debugging wire-based hardware is immeasurably more scary and difficult than debugging code.I first built out the I/O module of the circuit, followed by the first of the two 4-bit adder circuits. After a break for a few weeks I finalized the circuit and built the other 4-bit adder and redesigned the I/O modules so that Input and Output were on separate breadboards.

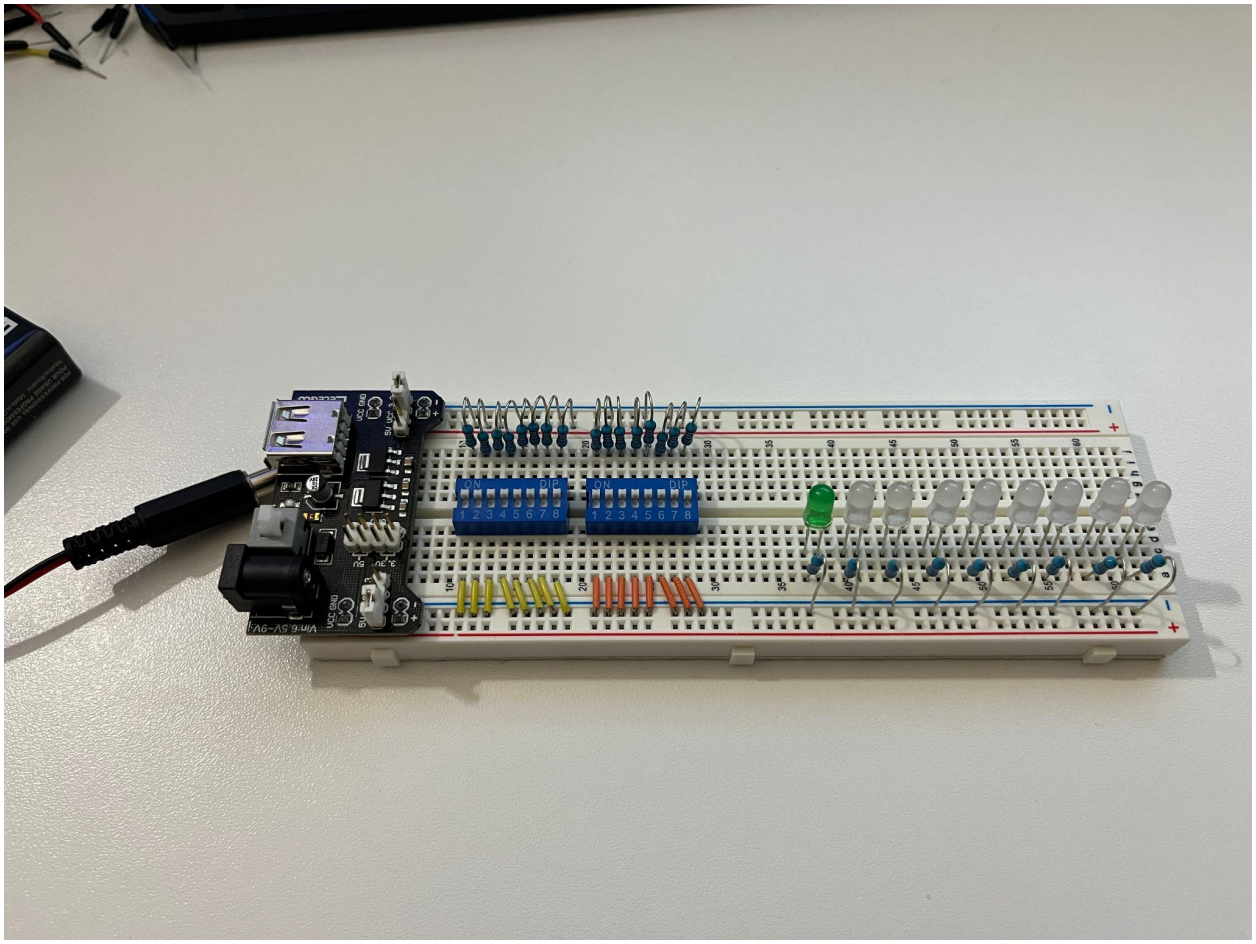

Diagram 1.

This is the original rough design of the I/O module when it was based on a single breadboard. You will see a slight change within the next few diagrams.

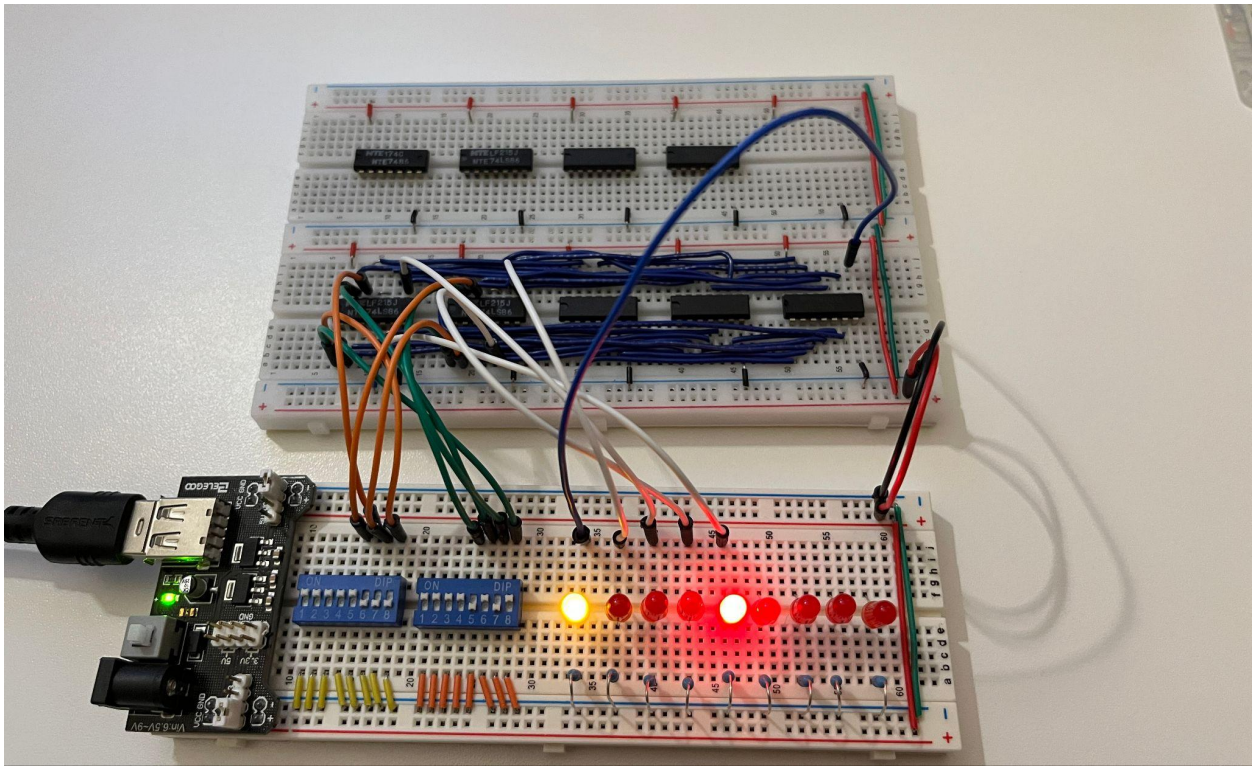

Diagram 2.

Here is the updated I/O module connected to the completed 4-bit adder. Here the yellow LED represents the carry out / overflow flag. Here the circuit is computing `1010 + 0111 = 10001`.

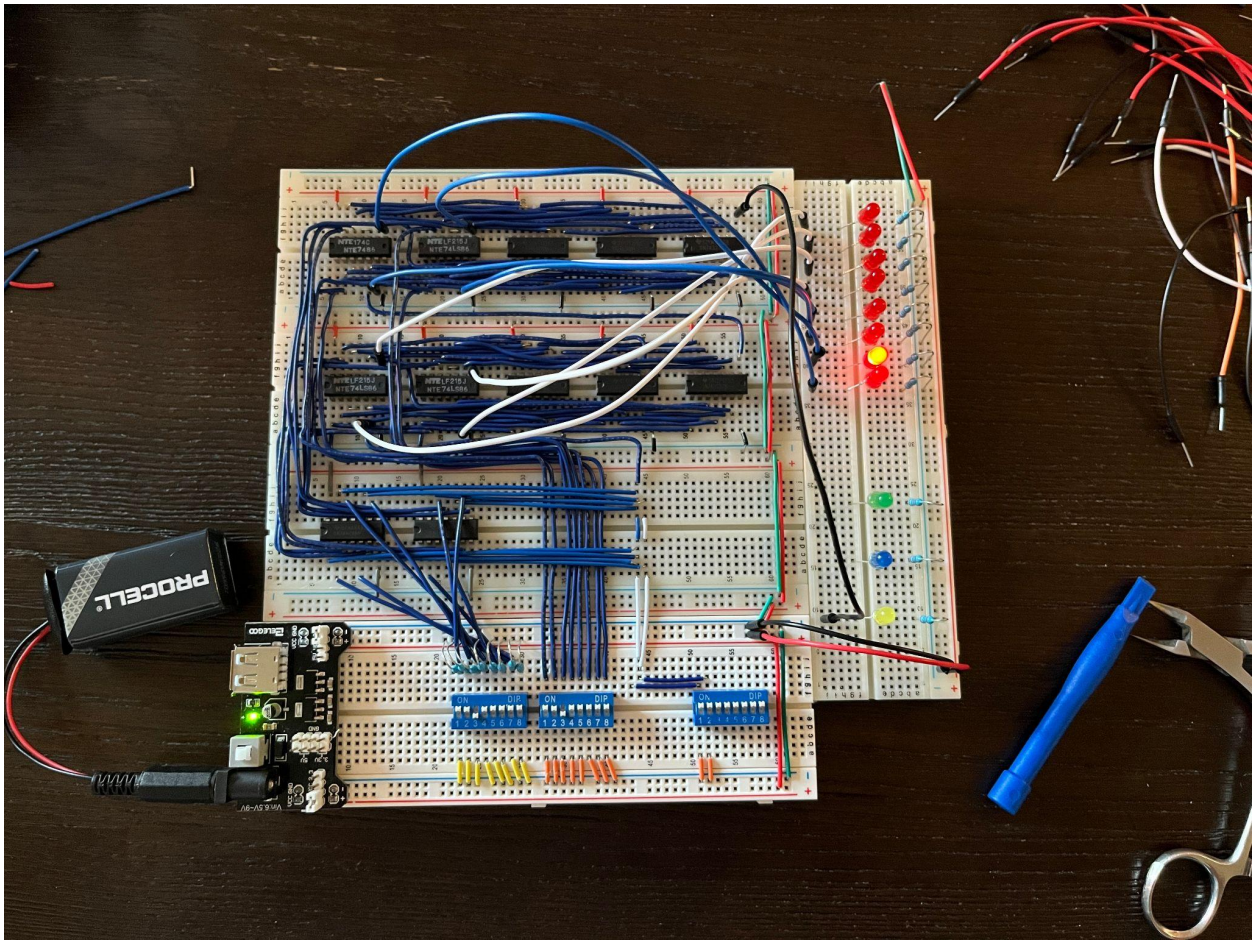

Diagram 3.

Here is the completed 8-bit adder fully wired together and connected to the new separate Input and Output modules. Here the circuit is computing `00100000 + 00100000 = 01000000`.

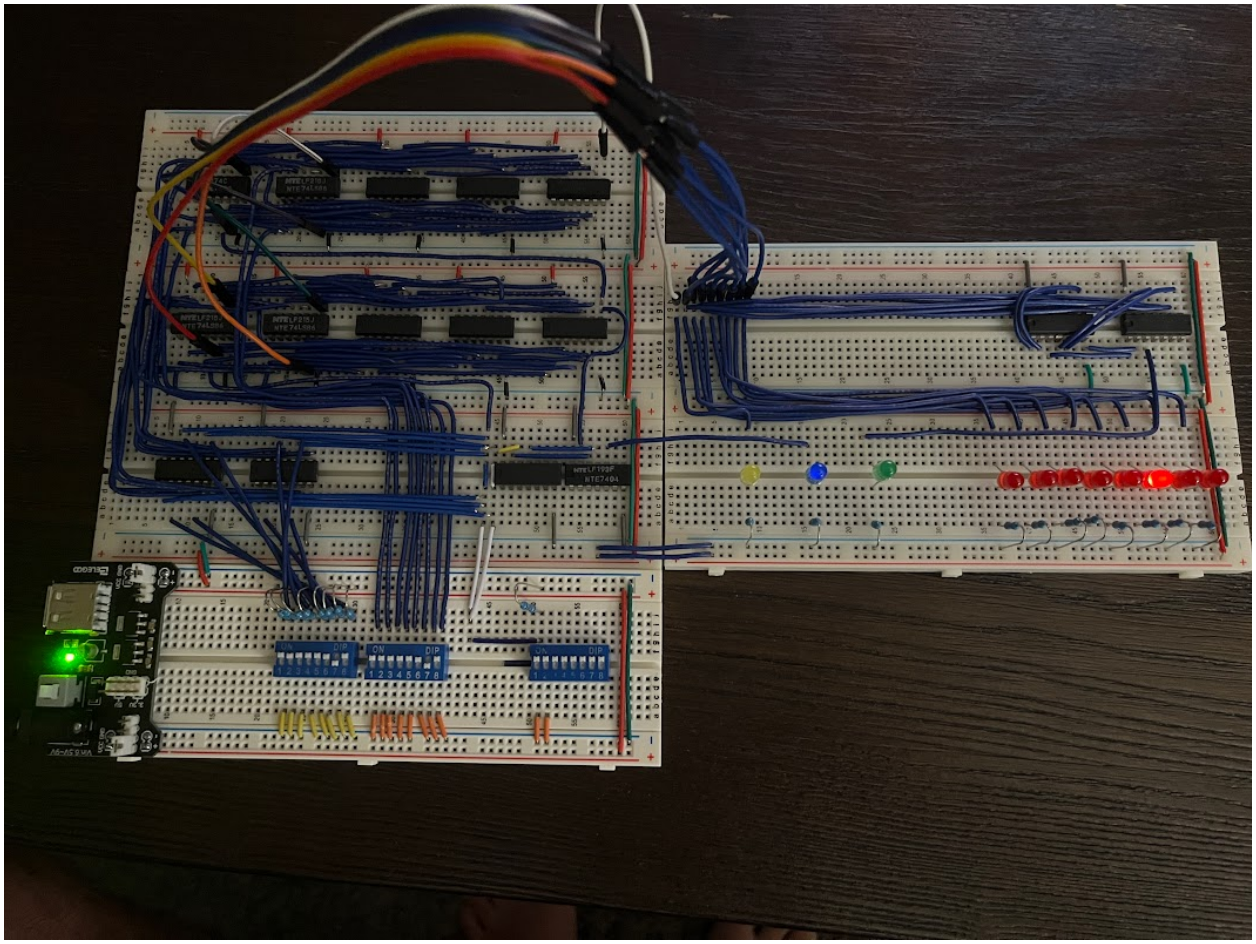

Diagram 4.1.

Here is the final circuit with its completed co-board that includes the output and the zero-checking for the binary output. Here it is computing `00000010 + 00000010 = 00000100.`

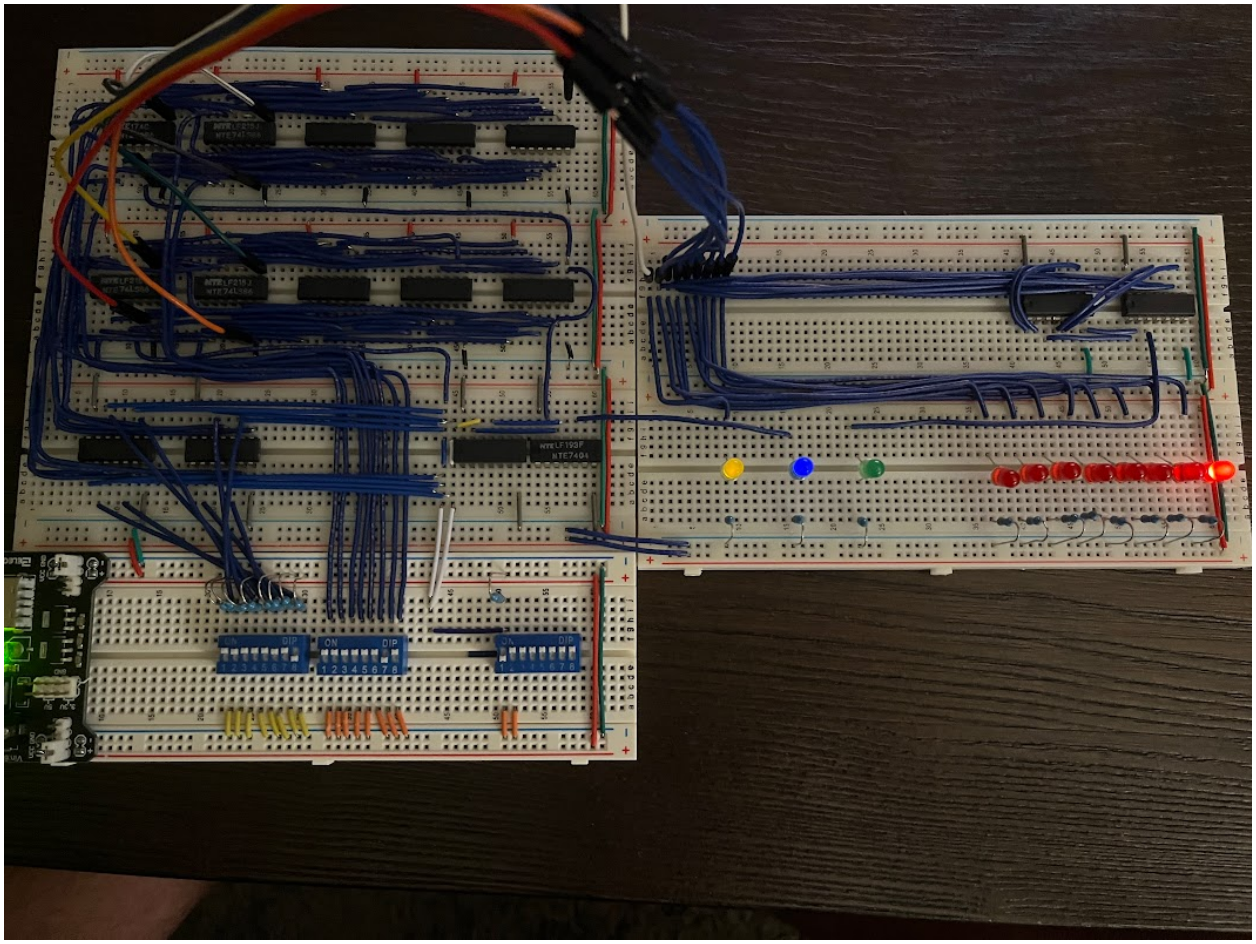

Diagram 4.2.

Here is the same circuit from diagram 4.1. but computing in the subtraction state. Here it is computing `00000001 - 00000010 = -00000001.`

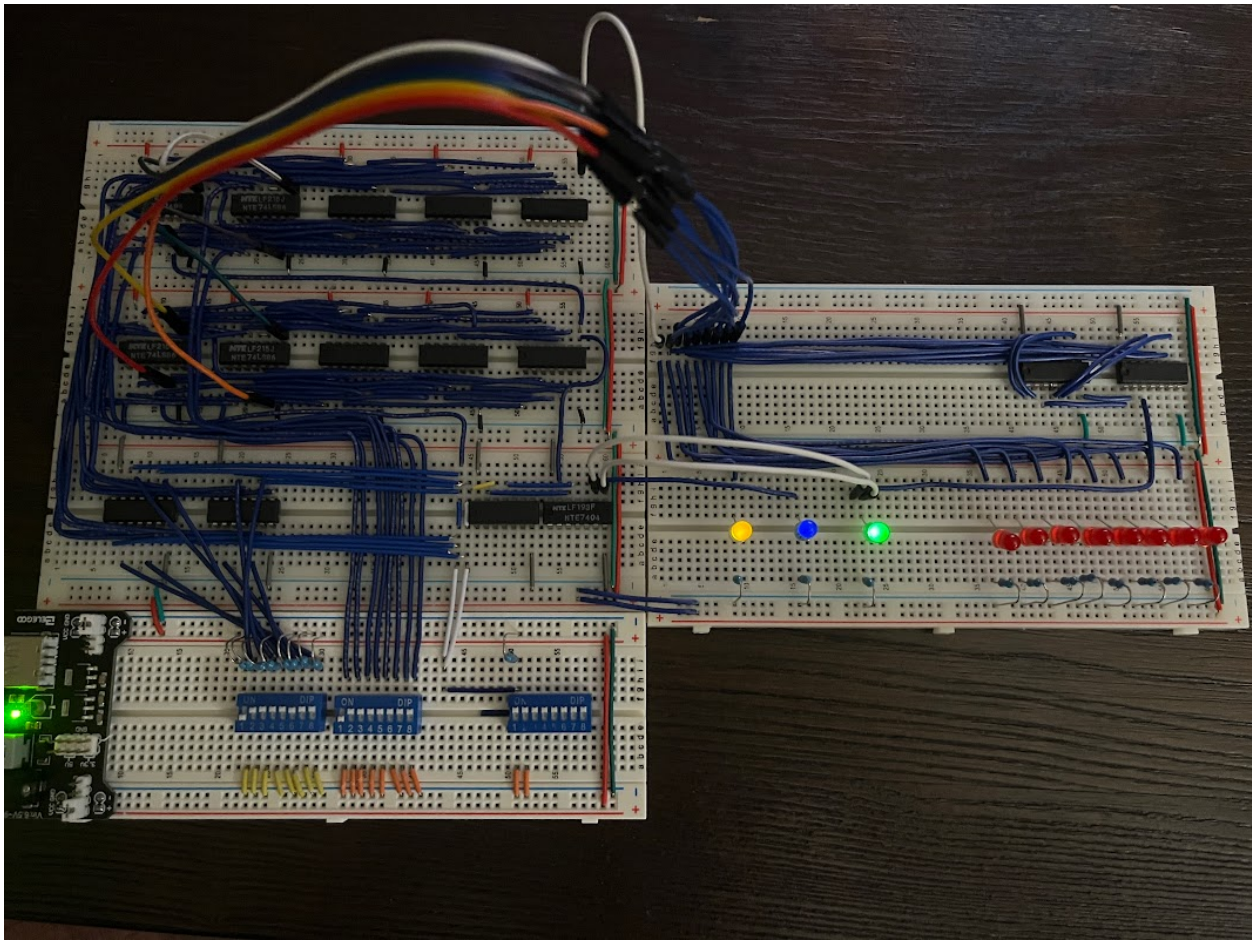

Diagram 4.3.

Here is the same circuit from diagram 4.1. But with an overflow output showing. Here it is completing an operation that overflows and triggers the overflow flag. Its computing `10000000 + 10000000 = 100000000.`

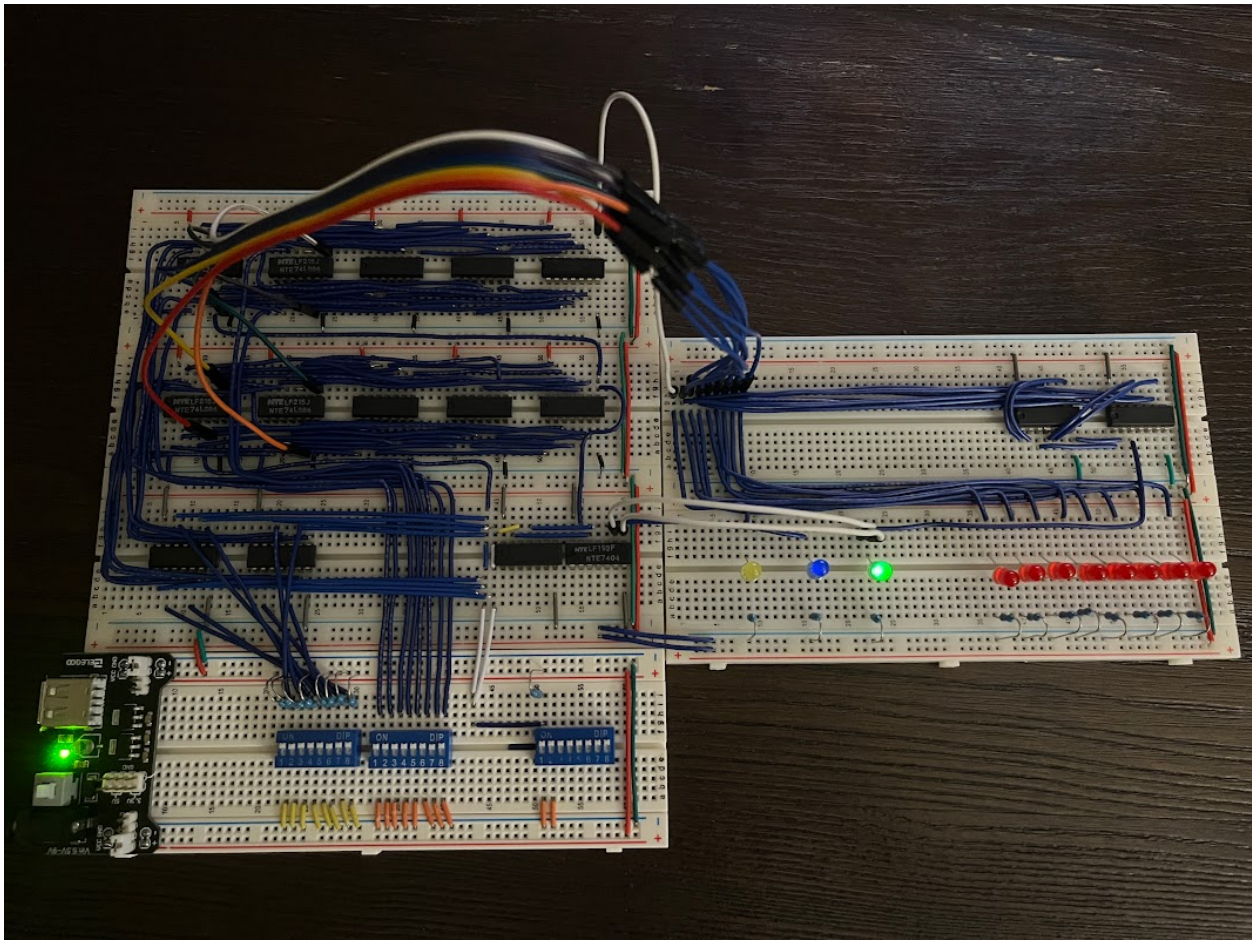

Diagram 4.4.

Here is the same circuit from diagram 4.1. But with zero input and is therefore triggering the zero input flag.

3. Components & Tools Used

3.1. Software Tools & Resources

- Logisim Evolution:
  - https://github.com/logisim-evolution/logisim-evolution
- Tinkercad Design:
  - https://www.tinkercad.com/dashboard
- Digital Computer Electronics Book:
  - https://archive.org/details/367026792DigitalComputerElectronicsAlbertPaulMalvinoAndJeraldABrownPdf1

3.2. Hardware Components

- 830 Point Breadboard: 7x
- 8-Input Dip Switch: 3x
- LED (Red): 8x

- LED (Green): 1x
- LED (Blue): 1x
- LED (Yellow): 1x
- Resistor (220Ω): 20x
- Patience: 3x
- 5/3.3V Elegoo Breadboard Power Supply: 1x
- 22 Gauge Solid Core Wire Spool: 2x
- 7404 TTL: 1x
- 7408 TTL: 5x
- 7432 TTL: 4x
- 7486 TTL: 6x

4. Limitations

This project has had a myriad of issues, limitations, and blocking problems with its development but I was able to power through. The most difficult part of this project was conducting the necessary research and self-learning required to sufficiently understand what is happening when I push buttons. Additionally, I had to learn the basics of logic circuit design, logic gates, TTL chip functionality and structure. Also, since I wanted to develop this project with physical hardwiring, I had to take multiple intro circuit electric circuit design courses on YouTube and LinkedIn Learning. The hardwiring of the TTL chips and properly understanding the changing in voltages, amps, and resistances and how they affect each other gave me the biggest challenges of all. In the end, the digital versions of the circuits in Logisim work as intended / designed but this final physical version does not. This is mainly due to the quality of materials and general improper conduction of electricity from minor defects in the circuit.

I have learned a lot through this project and hope to continue to improve upon this same project and possibly work out these funky wiring issues.