Four Bit Binary Calculator

Ahmad Malik
The Cooper Union
Digital Logic Design - ECE150
Professor M. Billoo

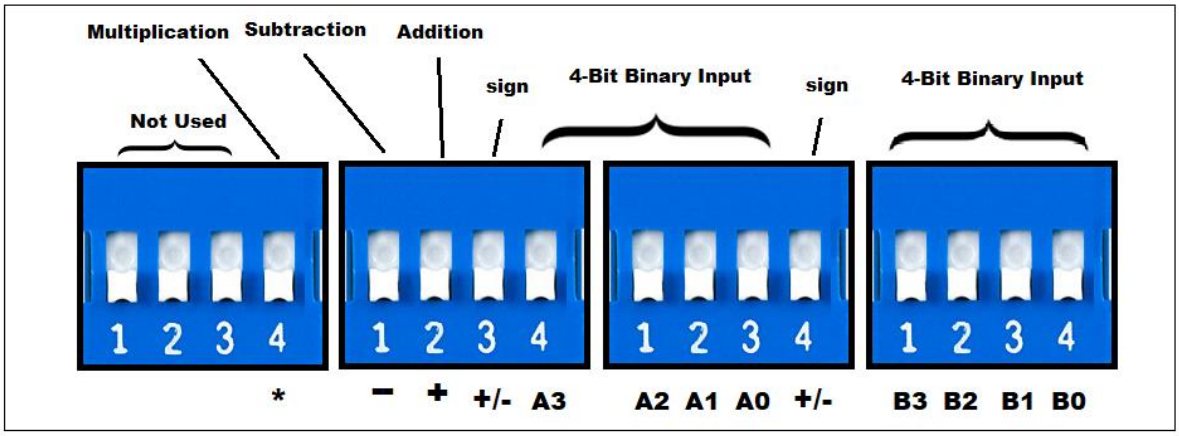
Introduction

The purpose of this project was to create a four bit binary calculator that could add, subtract, and multiply. The calculator should have two, four bit inputs that could undergo addition, subtraction, or multiplication. The way the output would be represented was our decision.

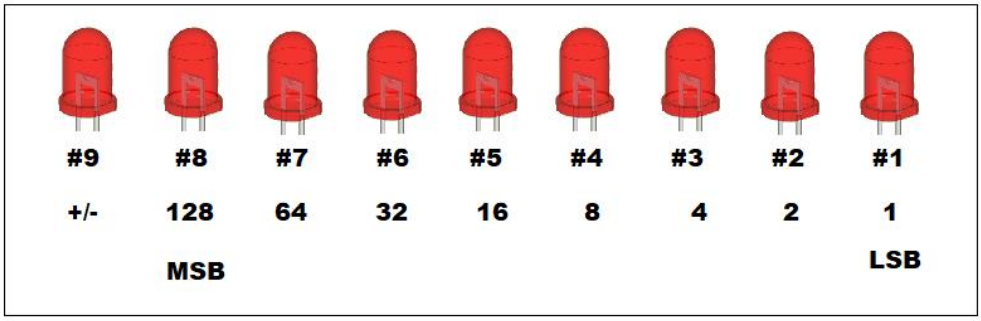
To represent the inputs, I chose to use four sets of 4-input DIP Switches, where a switch in the closed circuit position represents a Binary 1 (HIGH) and a switch in the open circuit position represents Binary 0 (LOW). For each of the two 4-bit inputs, 5 Dip switches will be reserved for inputting the data. The fifth switch of each 4-bit input will represent the sign, where a closed switch represents a negative value and an open switch represents a positive value. Three DIP switches will be reserved to interpret the requested operation: addition, subtraction, and multiplication. The remaining three DIP switches will be unused.

For the output, I chose LEDs to represent the HIGH and LOW states based on whether the LED is powered on or off, respectively. When the LED is on, it would represent a binary value of 1, and when it is off, it would represent a binary value of 0; LED#5 will normally represent binary data except when dealing with subtraction (its purpose will be specified later), and LED#9 will always represent the sign value when dealing with multiplication (positive=LOW, negative =High).

DIP Switch INPUT



LED OUTPUT



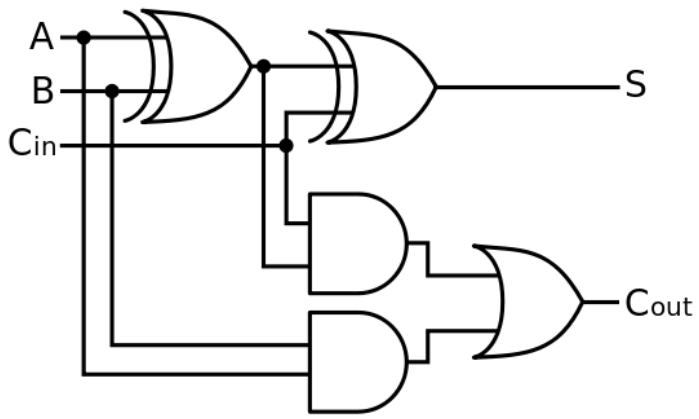
Theory and Design

Preliminary Considerations

We were only given the information that the calculator must add, subtract, and multiply. Whether the calculator would account for negative or positive inputs was unspecified, but our decisions had to be well documented. Since we were restricted in terms of electronic variety, LED's and DIP switches was the only way to display the input and output, with the exception of the 7-segment display which we hadn't learned how to operate. Our choices in IC chips were restricted to 2/4 input: XOR, AND, OR, NOR, and NAND gates. We were also supplied with 4-input hex inverter, which I considered using but ultimately decided not to (I'll cover why). Naturally, it was in our best interest to simplify and reduce the number of gates we used, with reason of course. Lowering the gates means less time spent wiring, less supplies used, and greater energy efficiency.

Addition is Key

Multiplication and addition are both additive processes. This means the bits are simply added linearly, and the carries are accounted for by adding them to the next column. Multiplication is a slightly tricky because between each bit addition, there is some bit comparison, and then a "bit shift". In class, we learned that a Full Adder is the simplest form of bit addition that accounts for any carries from the previous column.



INPUTS			OUTPUTS	
A	B	C_{in}	SUM	CARRY $_{OUT}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

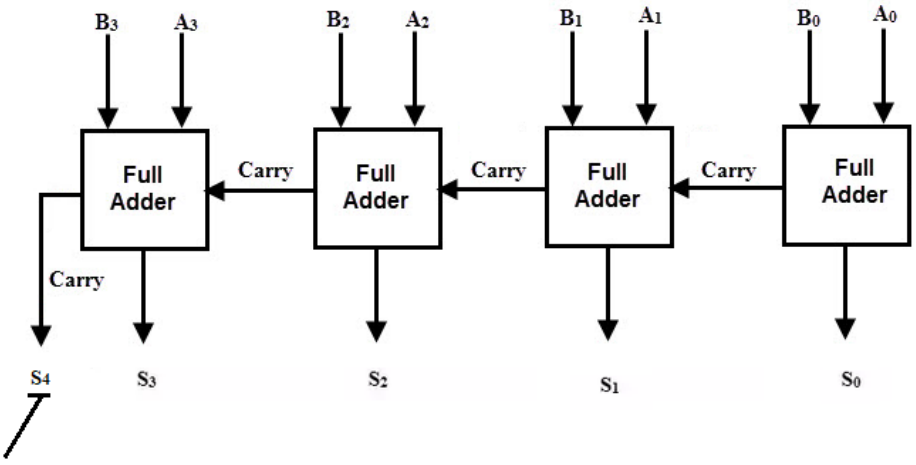
Above is a typical Full Adder. It takes two binary inputs along with any carries from any previous adders, and outputs the binary sum along with any carries that are issued in the process.

Subtraction is much more complicated because it requires borrowing from other columns which actively changes every involved column during the borrowing process. This becomes a very messy and unorganized process compared to simple addition. However, using the process of 2's complement, subtraction becomes an addition problem. Using 2's complement, the binary numbers are treated like the addition of a negative number; the first binary number is unchanged, and the second number is then negated and added. For binary numbers, converting a 4-bit number to its negative form requires inverting all 4 digits and adding one. Then the two binary numbers are added and the result is given in 2's complement. If the result is a negative number, the result will be in 2's complement, and taking the 2's complement of that result will yield the actual standard binary form; otherwise a positive the result in 2's complement is equivalent to its binary form.

Based on the universal application of addition, this calculator will primarily use Full Adders as its means of bit processing of all inputs, regardless of the operation.

Addition of Bits

For addition, a 4 bit input can have a maximum decimal value of 15. Thus, the largest sum of two 4-bit inputs is a decimal value of 31, which is a 5-bit output. Since bit addition is calculated by simply adding the two inputs and passing on the carries, it becomes an reiterative process that makes for the application of a full adder. When several Full adders are combined such that the carries tie them together, the result is a simple bit adder. The bit overflow in this setup is easily accounted because the final carry value of the sum will be the fifth bit.



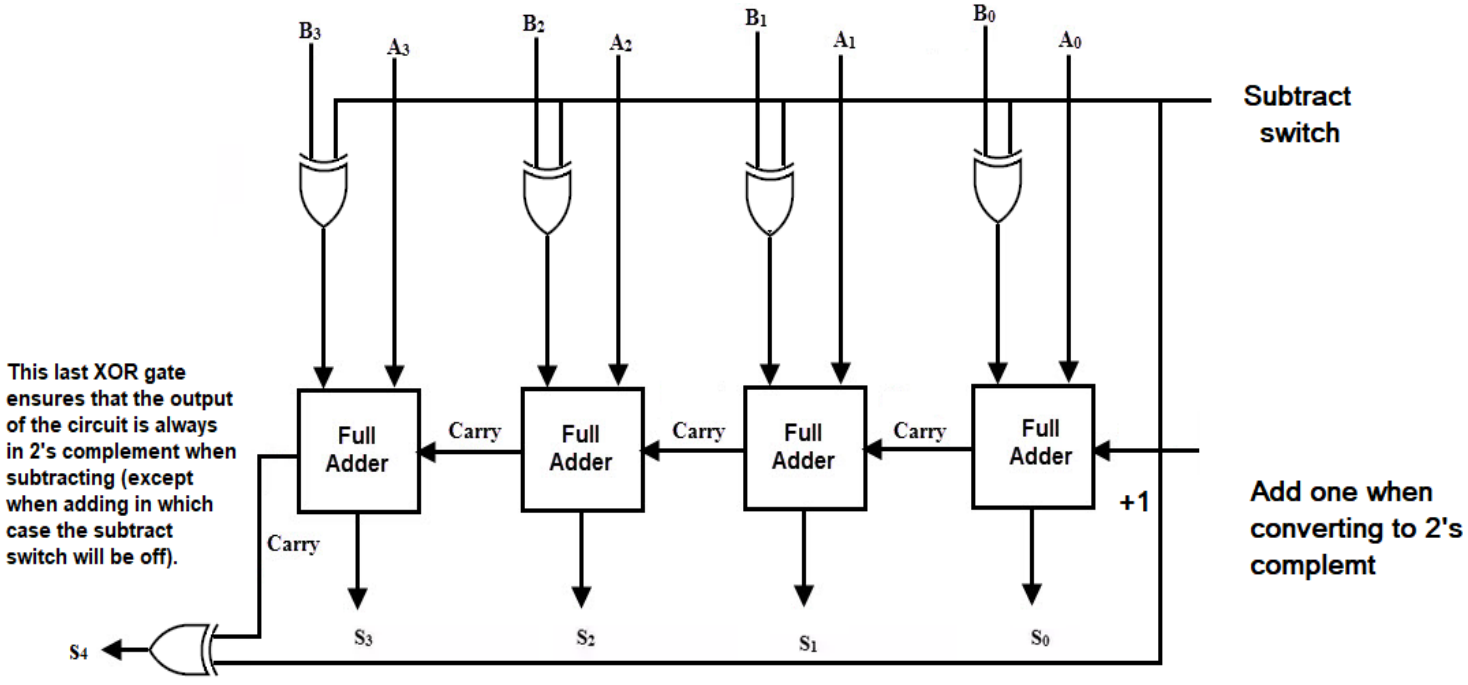
This last carry accounts for any overflow and outputs the 5th bit of the adder. If the output was resitricted to 4 bits, the carry would be ignored and the maximum decimal output of the system would be15. The additional carry bit nearly doubles that value.

Subtraction and Negative Inputs of Bits

For subtraction (and negative inputs in general), overflow depends on how I restrict the inputs. If A and B represent a 4-bit binary number, subtracting B from A yields the expression: A-B. If A and B are both positive, then the smallest output would be -15 and the largest would be 15, both of which are 4-bit in magnitude. If A and B are not restricted in polarity, the largest and smallest output would be 31 and -31, outputs of 5-bit magnitude. A 6th bit would be needed just to differentiate between the negative and positive results. This is because when taking 2's complement of decimal 15 (4 bits) to get its negative, the 15 is represented in 5 bits in 2s complement. When added to another negative value, say decimal 7 who's 2's complement is also 5 bits, the result is always going to be 6 bits. Therefore, restricting the inputs to be positive when subtracting keeps the output from flowing about 5-bit.

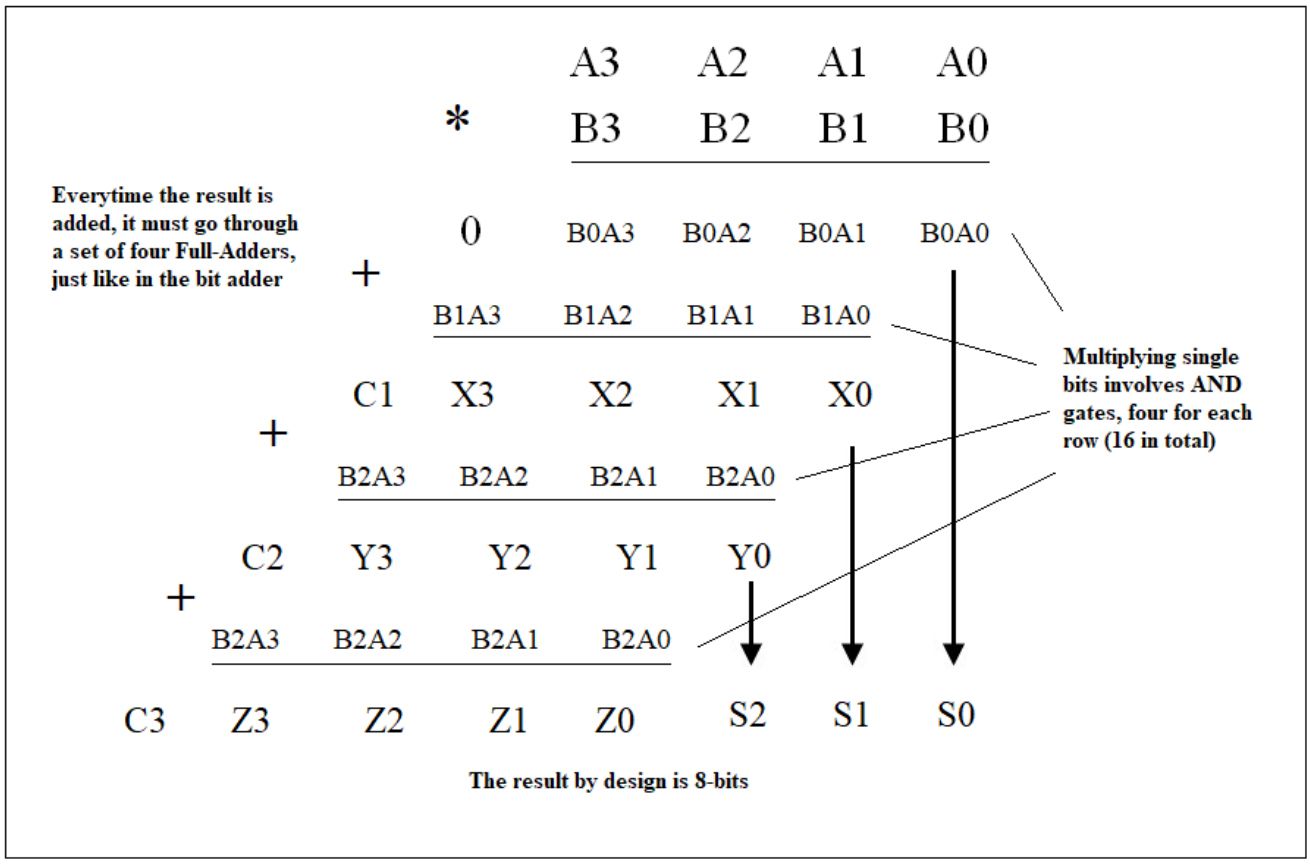
For my circuit, I have decided to keep both inputs positive because of bit overflow as mentioned above, and because incorporating signs would require a much larger circuit. Whenever an input would have a negative sign, the circuit would always have to convert it to 2's complement. Every time this would happen. The input would have to be inverted and added 1, which would require gates to invert the input or a hex inverter, along with five whole full adders just to account for the added 1 and the extra bit. This would be needed for both inputs A and B. Because of these reasons, I've decided not include signed inputs for both addition and subtraction.

I am also not going to use a hex inverter because although it would swiftly invert the inputs of B when subtracting, there wouldn't be an efficient way to toggle it on or off such that the input would not be inverted with a push of a switch. Instead, a few XOR gates would do the trick with all the functionality I would need. Additionally, this circuit can then be integrated with the original adder because toggling the subtract switch would enable/disable normal bit addition.



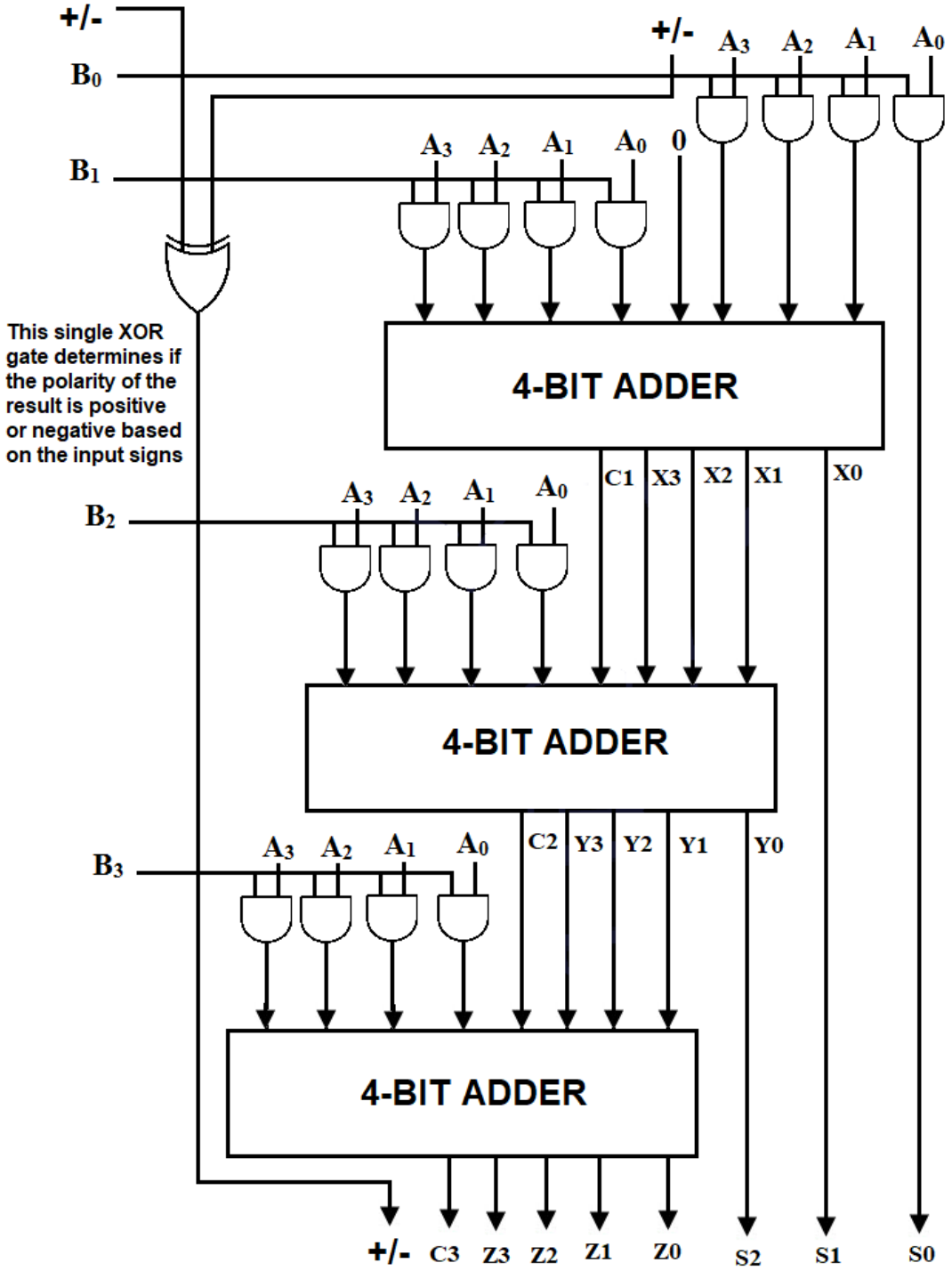
Multiplication of Bits

As mentioned before, multiplication and addition are both iterative processes. Addition is simply adding inputs and carries, and multiplication involves comparing the inputs and caries along with a summing of the compared bits. The comparing of bits in terms of bit multiplication is recognized as “AND” ing because when multiplying a single bit with another bit, the binary result is exactly the outputs of an AND gate. The summing of the compared bits between each step is the equivalent process of adding two 4-bit inputs as described in the adder. Thus, Full-Adders and AND gates will be the main logic involved in multiplication. Below demonstrates how bit multiplication works and where and when each operation takes place. (See next page for block diagram)



In terms of bits, a 4-bit input multiplied with a 4-bit input has a maximum decimal value of 225, which is an 8-bit output. My calculator will not only include an 8-bit output, but it will also carry an extra bit to show polarity. I made this decision because if the inputs when being multiplied have positive and negative signs, the method to determine the output polarity only requires a single XOR gate, unlike with addition/subtraction where signed inputs would require several more stages of Full Adders and inverting gates.

4-BIT MULTIPLIER



Controlling the Power and Operation

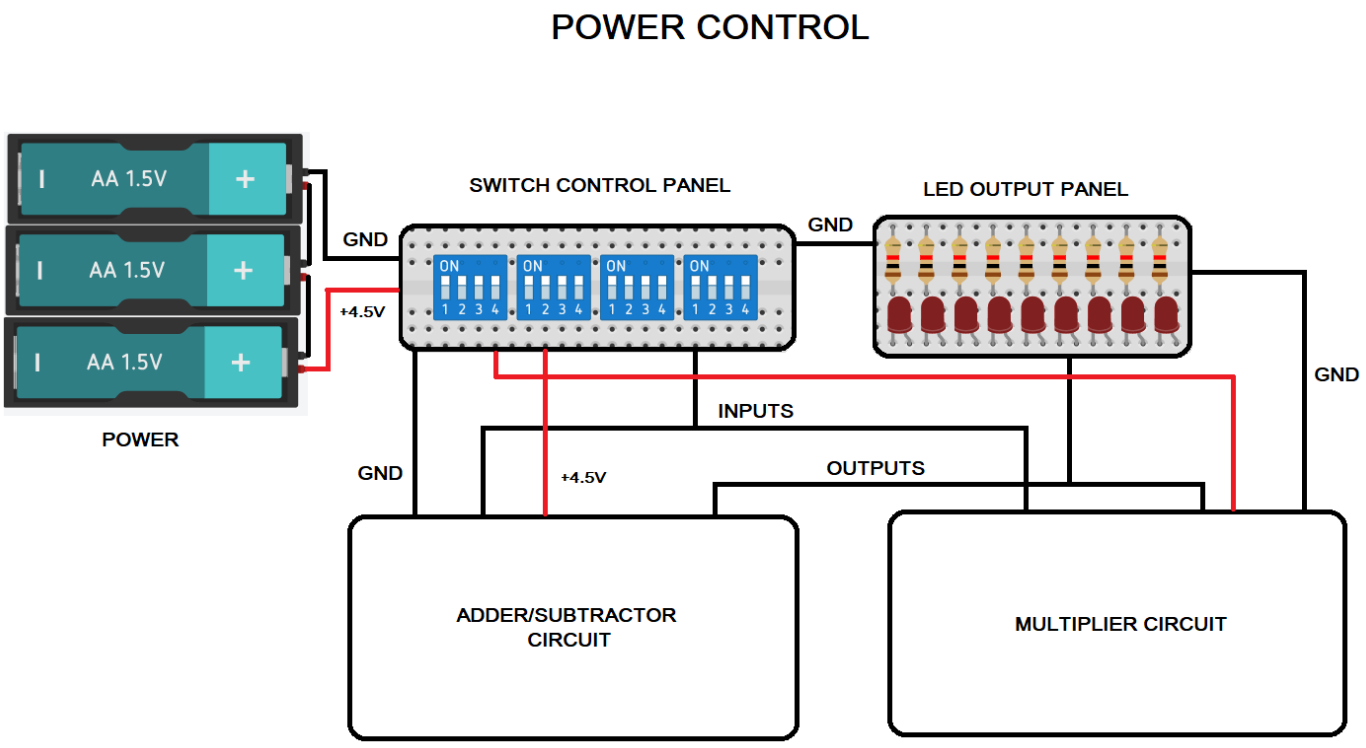
The user interface for my design involves a single row of DIP switches for inputs and a single row of LEDs for the outputs. This makes the design simple and easy to use for anyone familiar with binary.

Since all three circuits are integrated together into the same input and output line, there needs to be a way to control which circuit is operating so that outputs don't overlap.

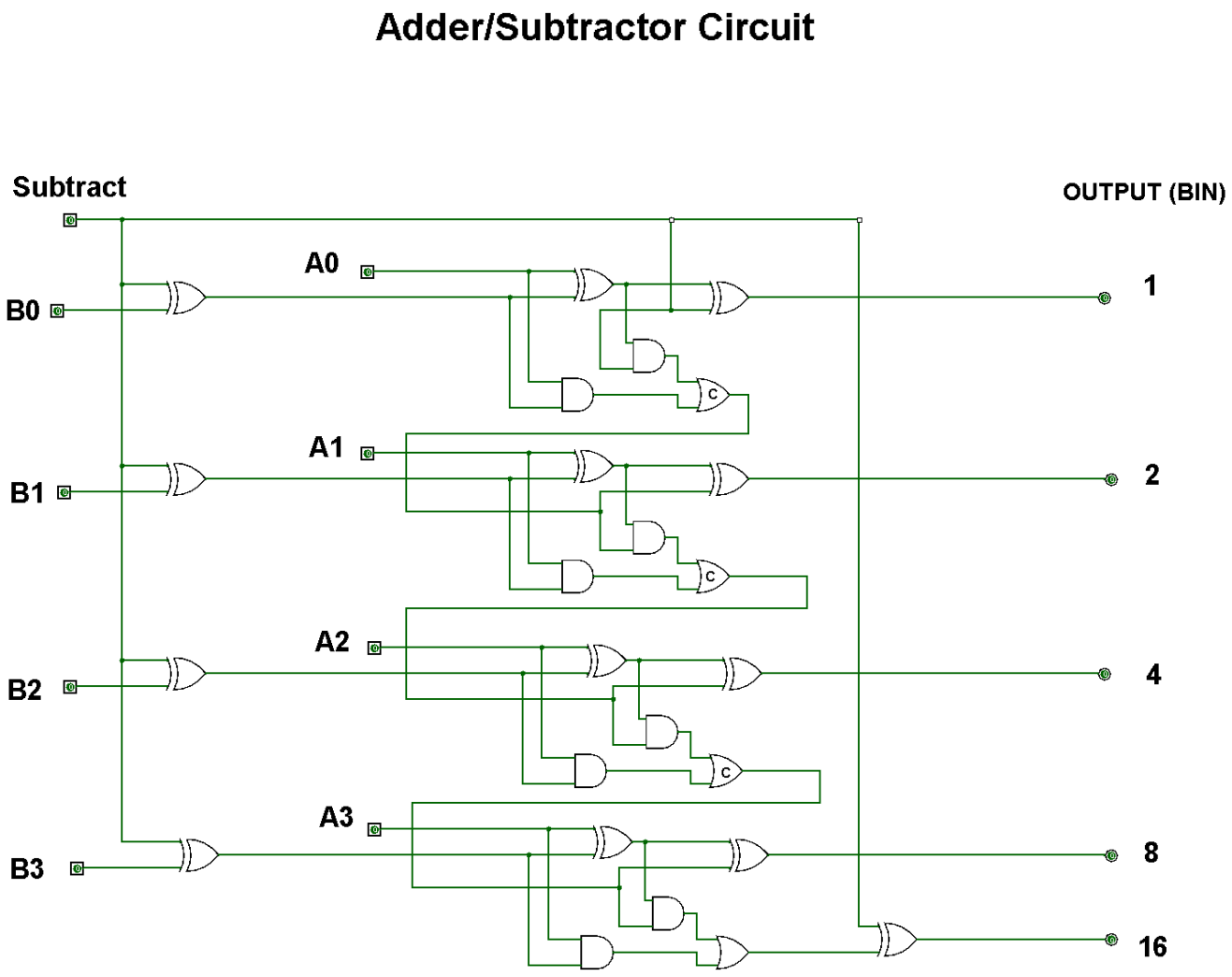
For addition and subtraction, they share the same circuit so they will occupy a single breadboard. The switch that enables addition will be responsible for powering the entire breadboard of the adder/subtractor. If addition switch is in the "ON" position, the board is powered and performs addition. If both addition and subtraction switches are in the "ON" position, the board will be powered and perform subtraction. If both switches are in the "OFF" position, the board will not be powered and won't trigger any outputs.

For multiplication, the breadboards that deliver the output will be wired separately from the adder/subtractor. The multiplication DIP switch will control the power of the multiplication circuit (ON=power, OFF= no power). To perform multiplication, both subtraction and addition switches must be off and the multiplication switch must be on to display the outputs.

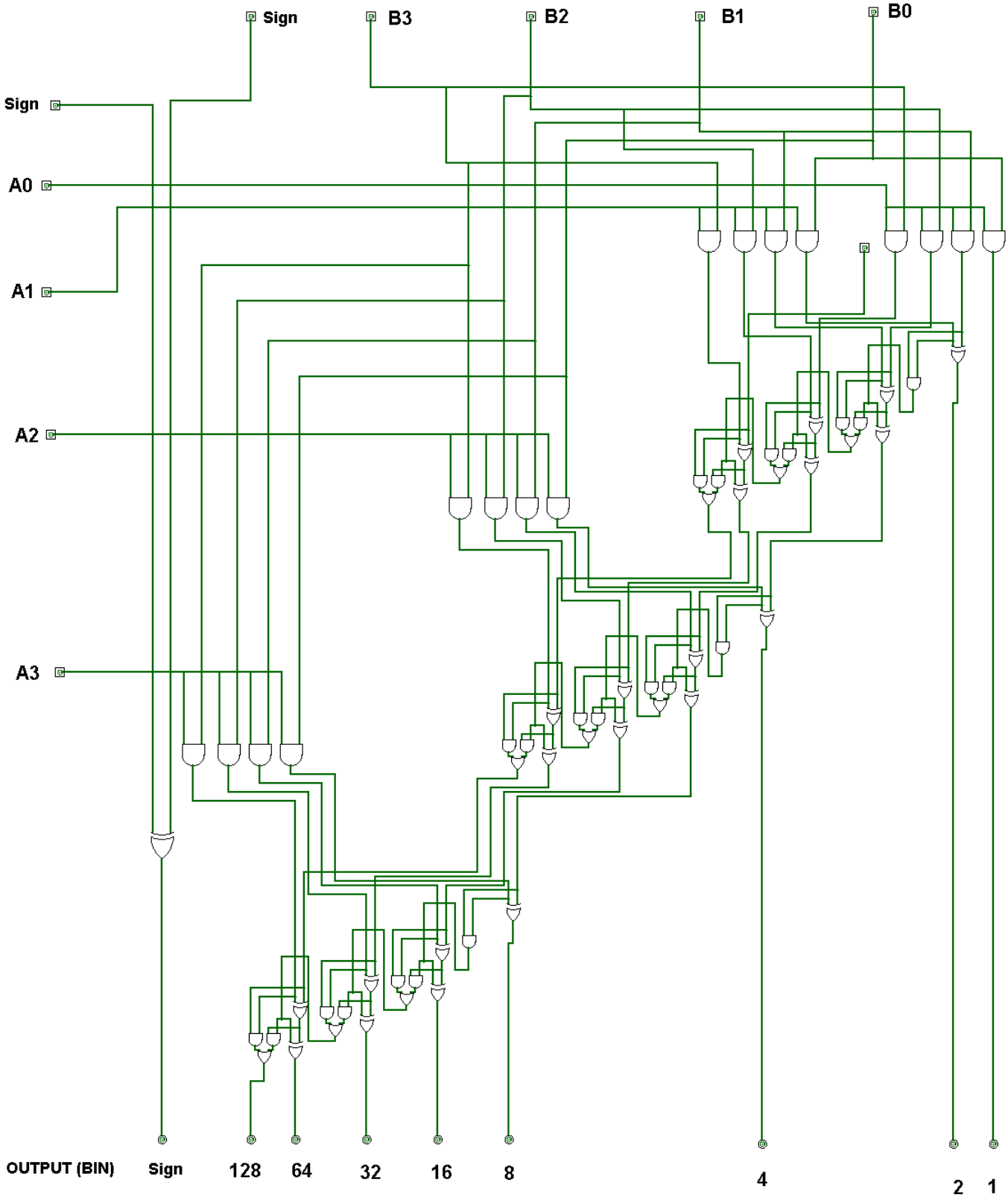
By isolating the circuits, none of the outputs and inputs should interfere with each other.



Gate Layout in Logism

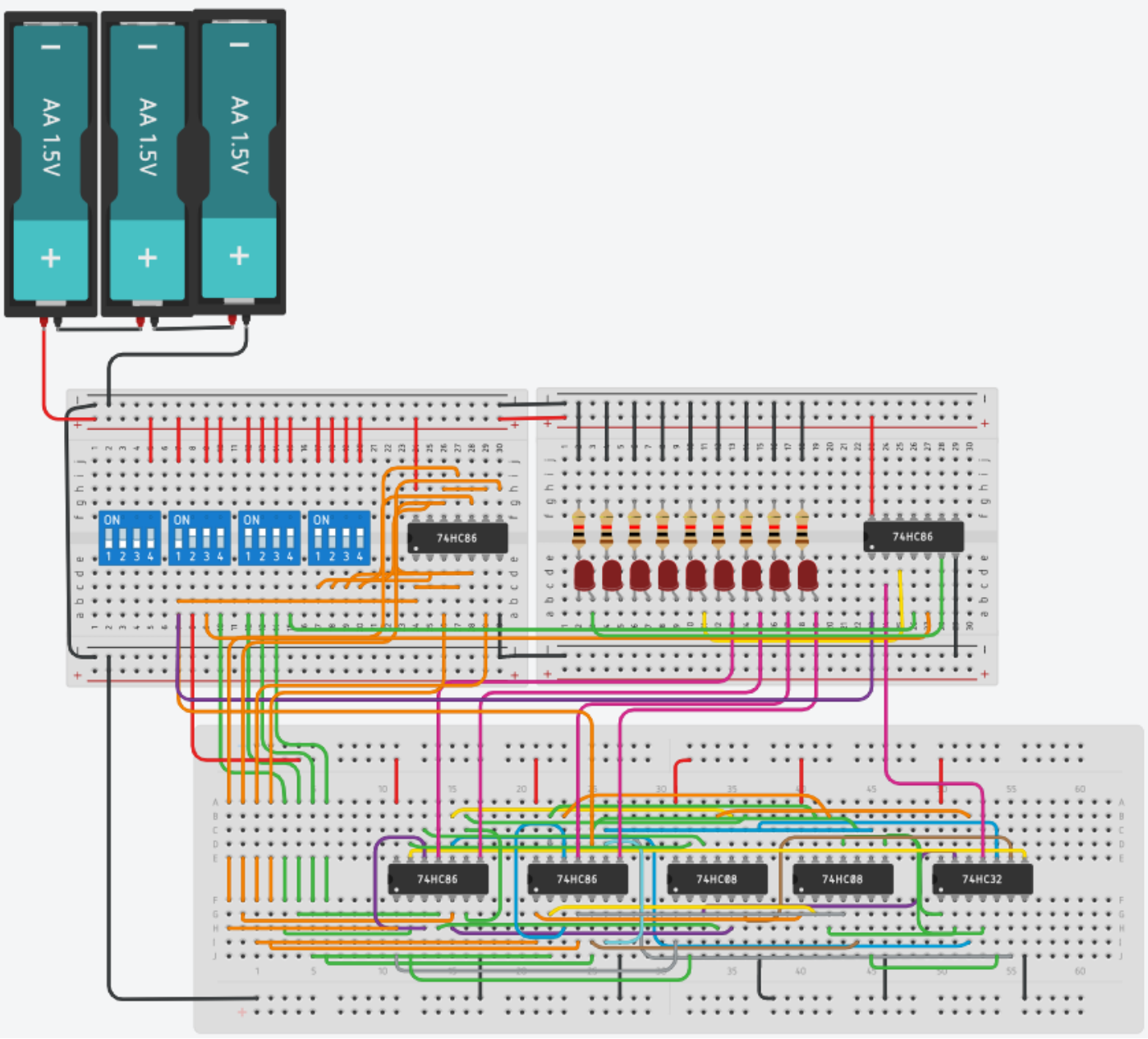


Multiplier Circuit

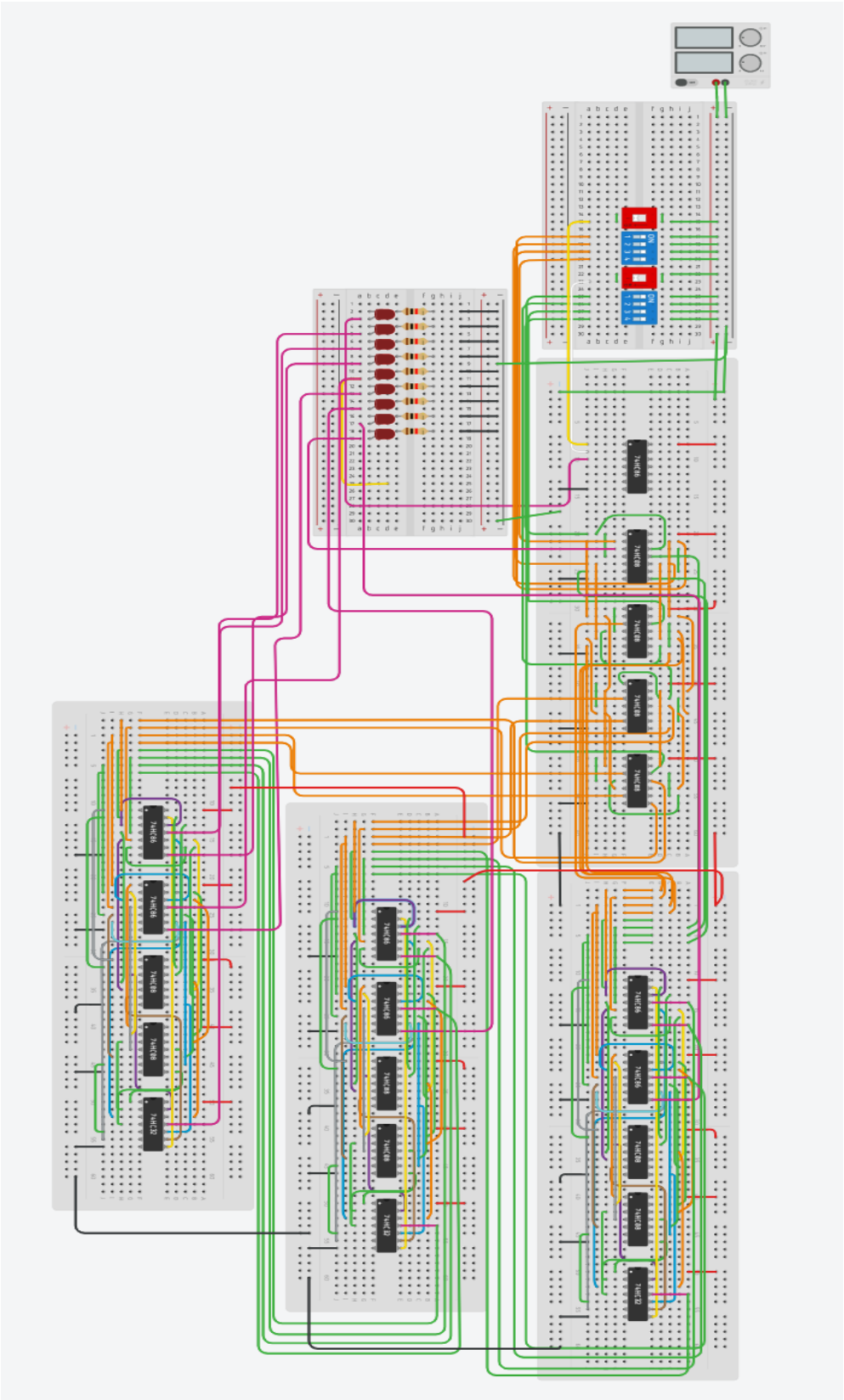


TinkerCad Simulations (in progress)

Adder/Subtractor Circuit



Multiplier Circuit



Conclusion

This project has helped me understand the applications of simple logics gates and how they can be used to make large and complicated circuits. When I first began this projects, I was very lost because I wasn't sure how multiplications and subtraction can be done using just transistors. But after learning about the Full Adder and 2's complement, the idea of a 4-bit calculator wasn't as abstract as it initially had seemed. Learning about how a simple full adder can be repeatedly used to make complicated circuits leads me to believe that the highly complex computers, phones, and technology in their bare bones stem from very simple circuit modules. I hope to accurately understand this someday.

Right now I am working on combining the Adder/Subtractor circuit with the Multiplier circuit on tinkercad. It is quite difficult because logic gates are highly sensitive to current and voltage, and I keep tripping the wrong gates. I plan to add more resistors to lower the chances of overloading the data pins. I feel that current might be passing into the wrong circuit boards through the data pins, which makes me wonder if diodes would help prevent that.

Overall, this challenging project has helped me grow my understanding of how simple transistors and logics gates are the foundations of complex machines.