

Report for Computer Systems Architecture

BROUGHT BY YADA YADA



**A comprehensive report on
our group project: a calculator
built on CircuitVerse.**

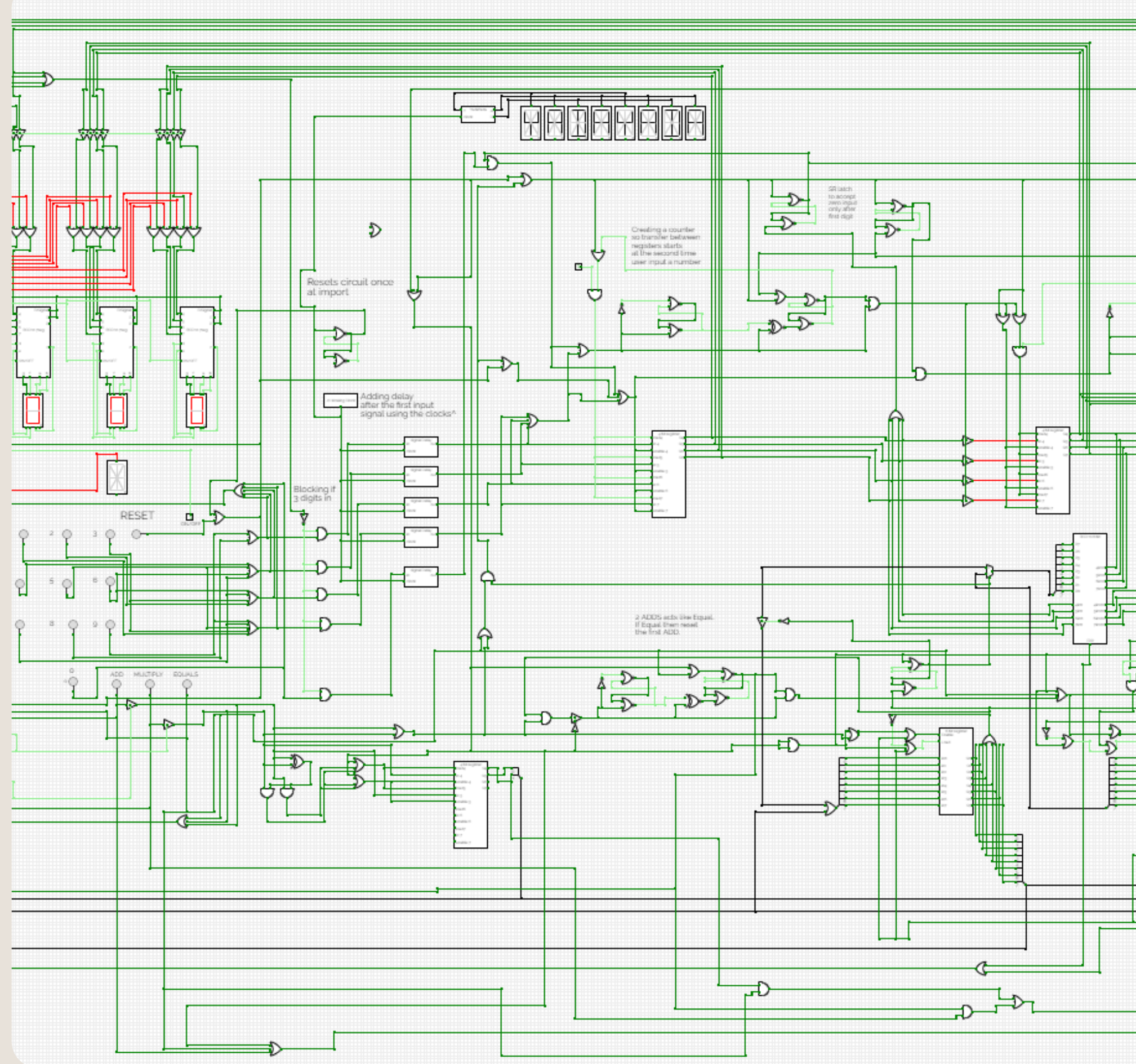
TABLE OF CONTENTS

About Our Project

How It Works

Sub-Circuits

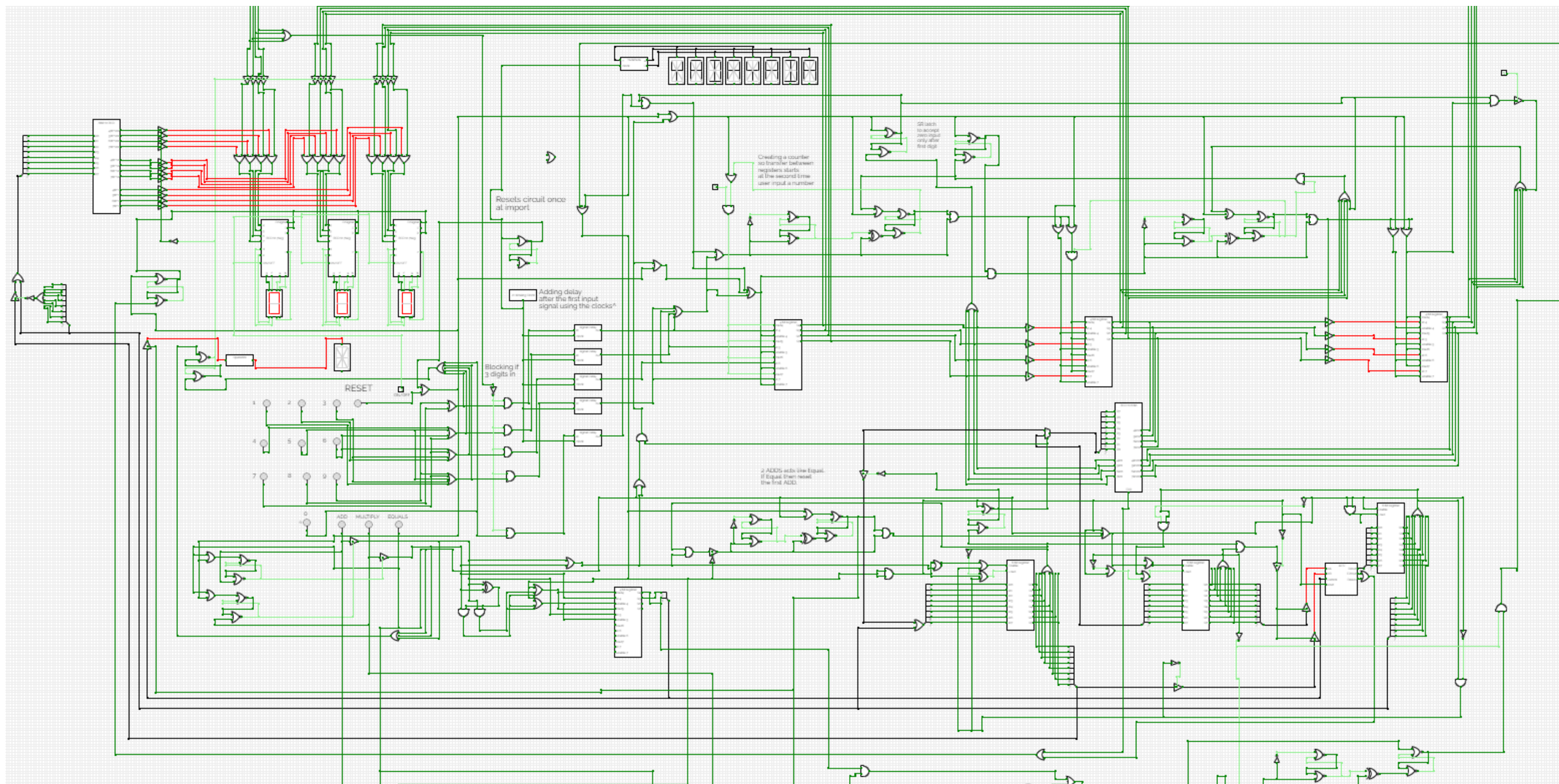
Contact Us



ABOUT OUR PROJECT

We were tasked to create a calculator on CircuitVerse using logic circuits. Said calculator must be able to perform addition and multiplication of binary numbers and display the result, as well as clear its own memory at the press of a button.

This is what we came up with..

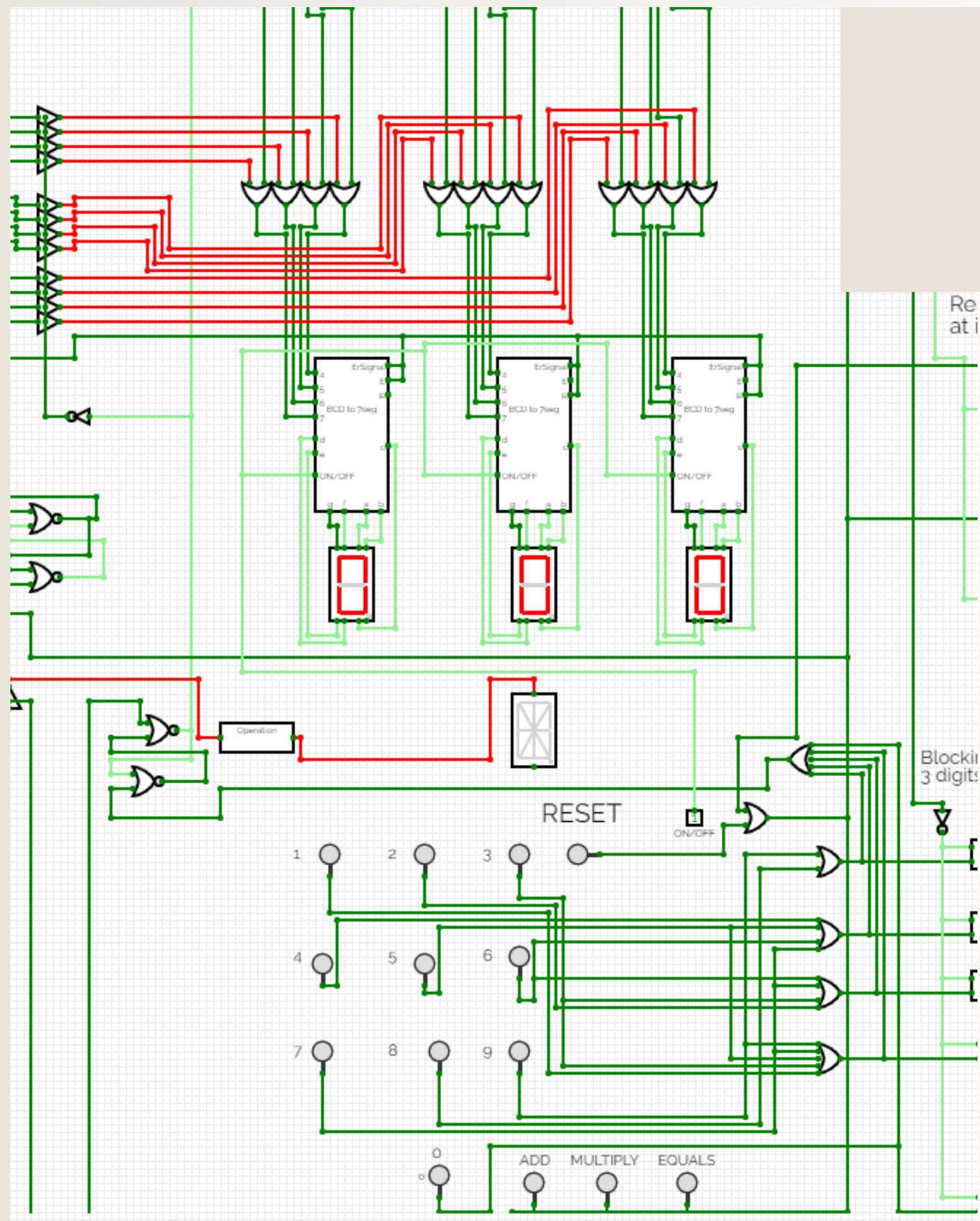


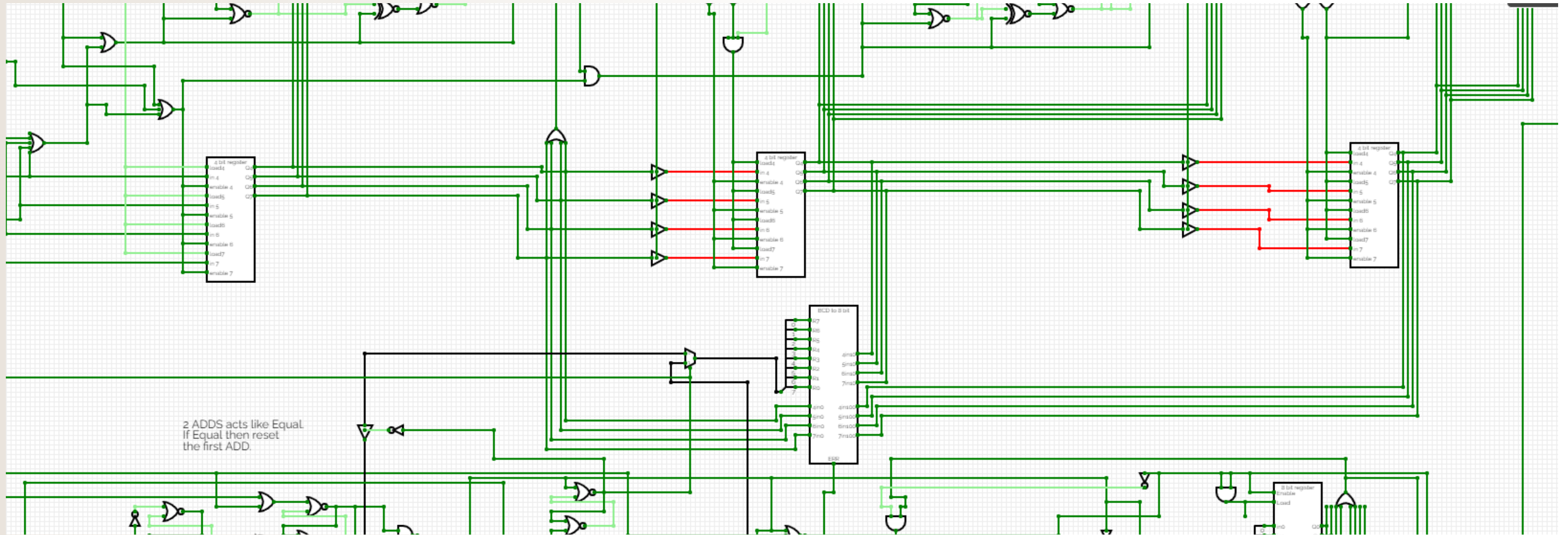
HOW IT WORKS

Circuit accepts user input through buttons (0-9). The input is displayed on three 7-segment displays, with the rightmost display always showing the number that was last pressed.

A 16-segment display shows the currently pressed operation (add or multiply).

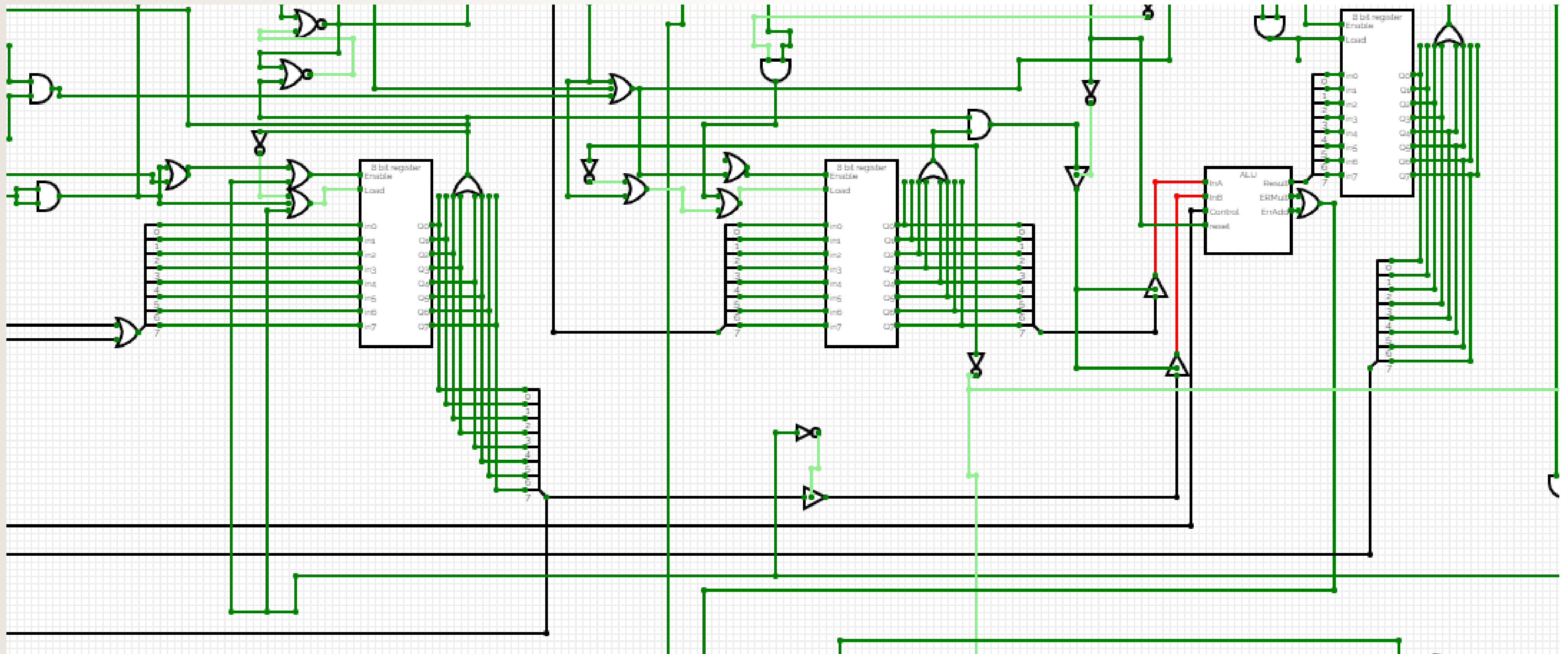
The user also has the option of turning the displays on or off, as well as resetting the calculator.



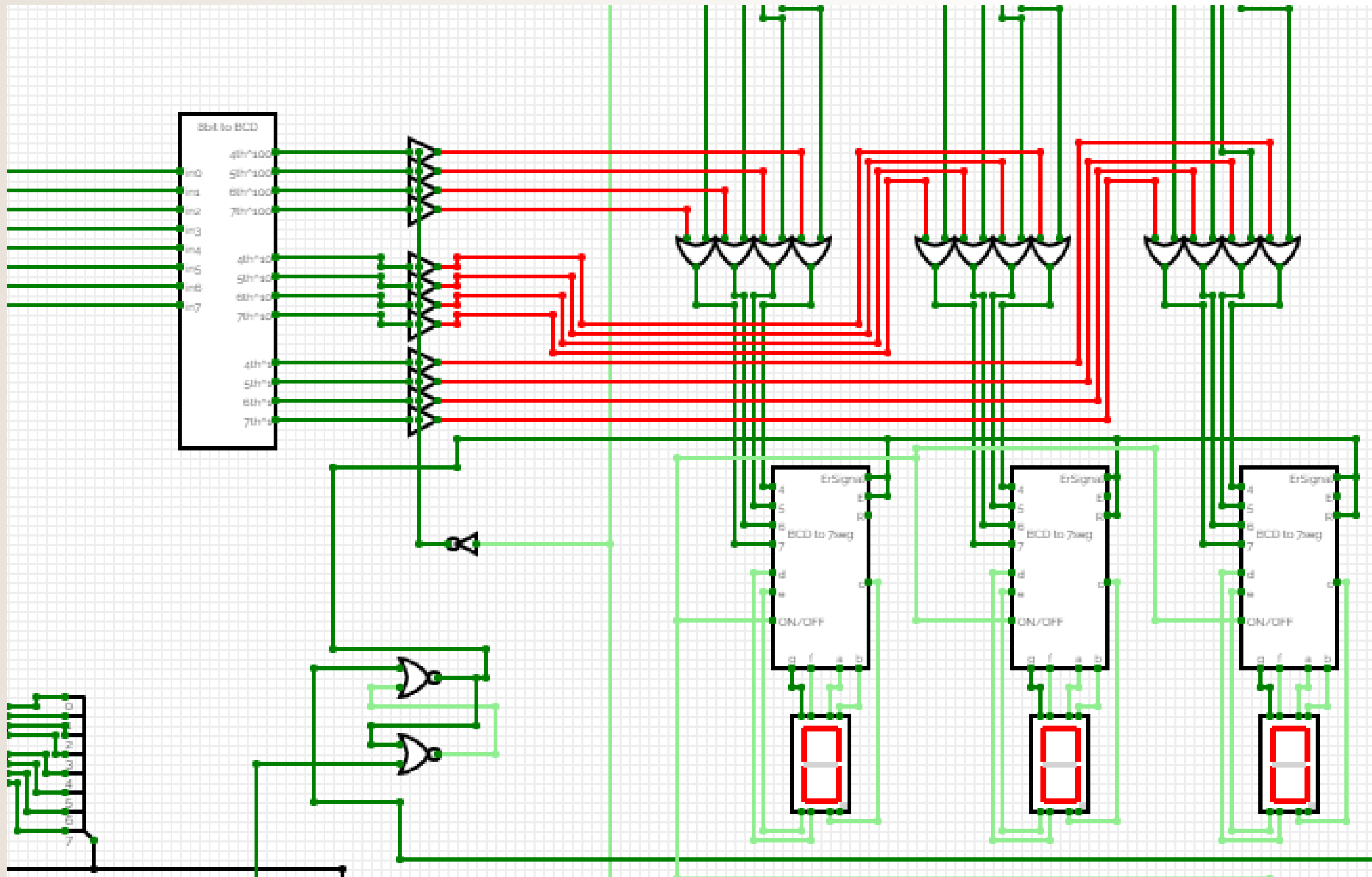


The data then passes through a set of 8-bit registers, with a lock-system in place, moving from every register into the next, in order to ensure that only the last inputted number will be displayed in the rightmost display.

We now need to convert these 3 numbers into one 8-bit number, and accept the next input.



The two 8-bit numbers that are now available to us are saved into separate registers. These numbers, as well as the operation signal itself, now pass through the ALU, which performs the operation (addition or multiplication). The result of that operation is saved into a separate 8-bit register.



One final conversion will now provide us with up to 3 8-bit numbers, one to represent the hundreds, the tens, and the ones.

These three numbers are now displayed again in the 7-segment displays.

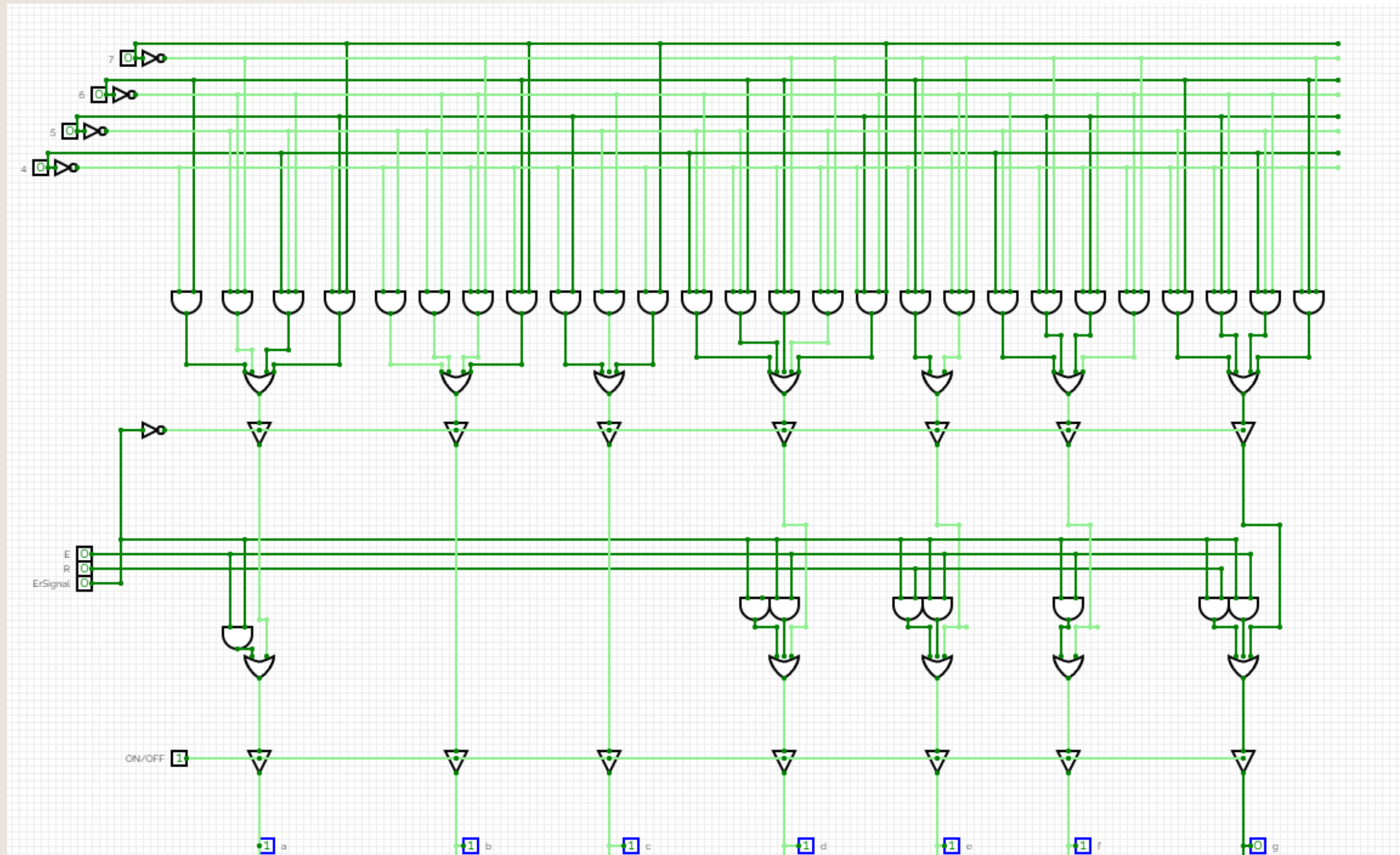
The user may now perform another operation or reset the circuit.

SUB-CIRCUITS

Let's take a closer look at the individual parts of the circuit:

- BCD to 7-segment Displays
 - Operation
 - Registers
 - Signal Delay
 - JK Slowing Clock
 - BCD to Binary
 - Half Adders and Full Adders
 - Multiplier
 - Divider
 - Subtractor
 - ALU
 - Binary to BCD Decoder
-

BCD to 7 Segment



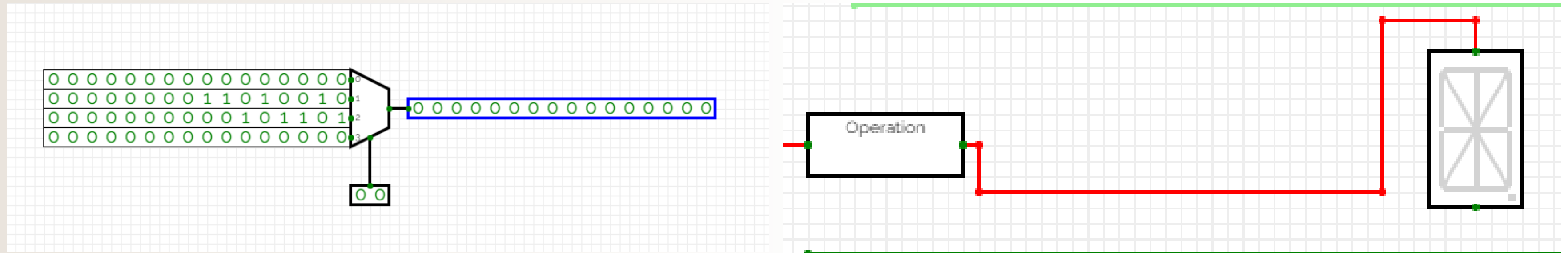
We had to find a way to display all input/result number combinations, as well as an error indication ('Err').

We mapped out which segments need to be activated in order to display each of the numbers 0 through 9 - for this purpose we used Karnaugh maps.

In addition, we created predefined 'E' and 'r' indications in case of invalid input/output.



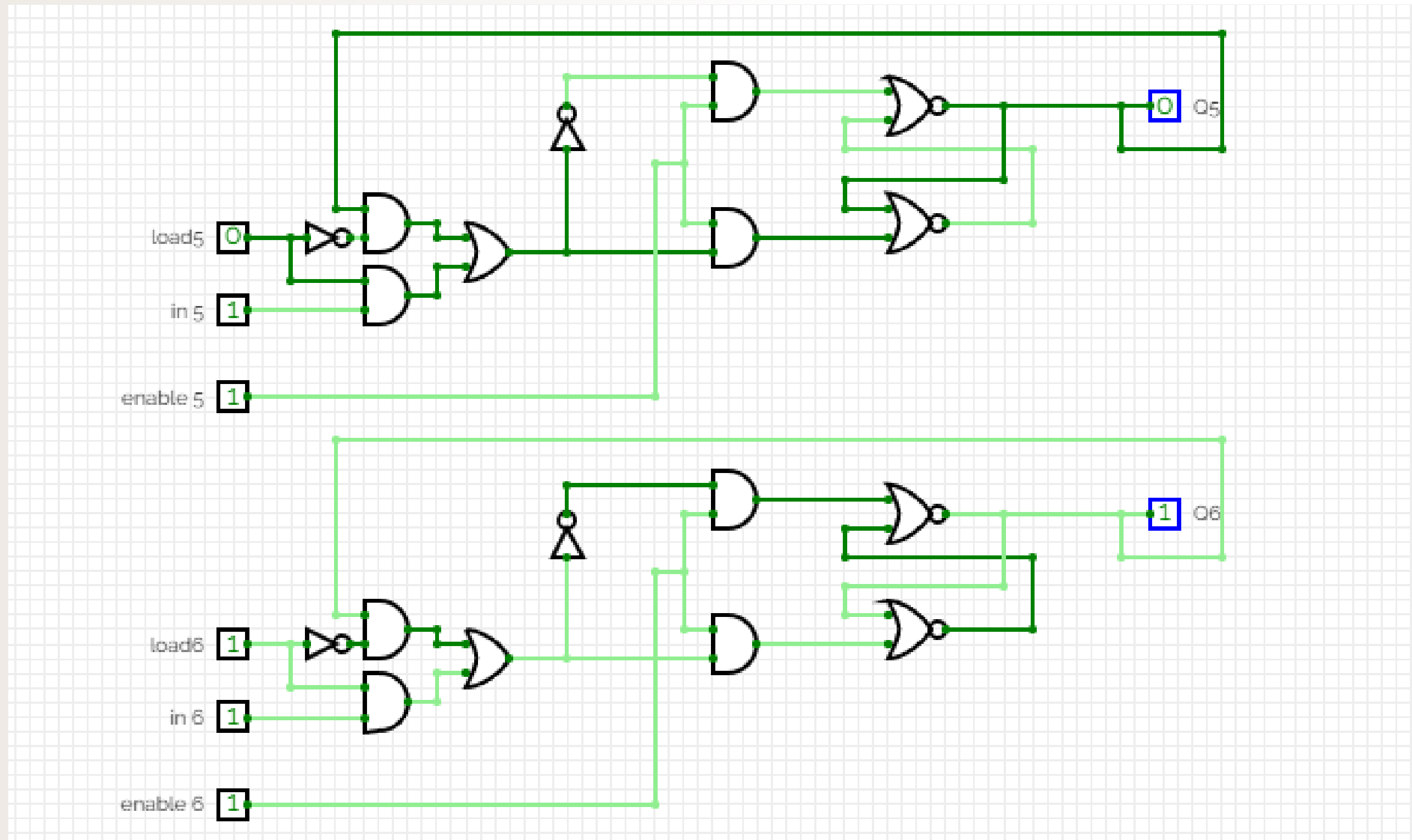
Operation



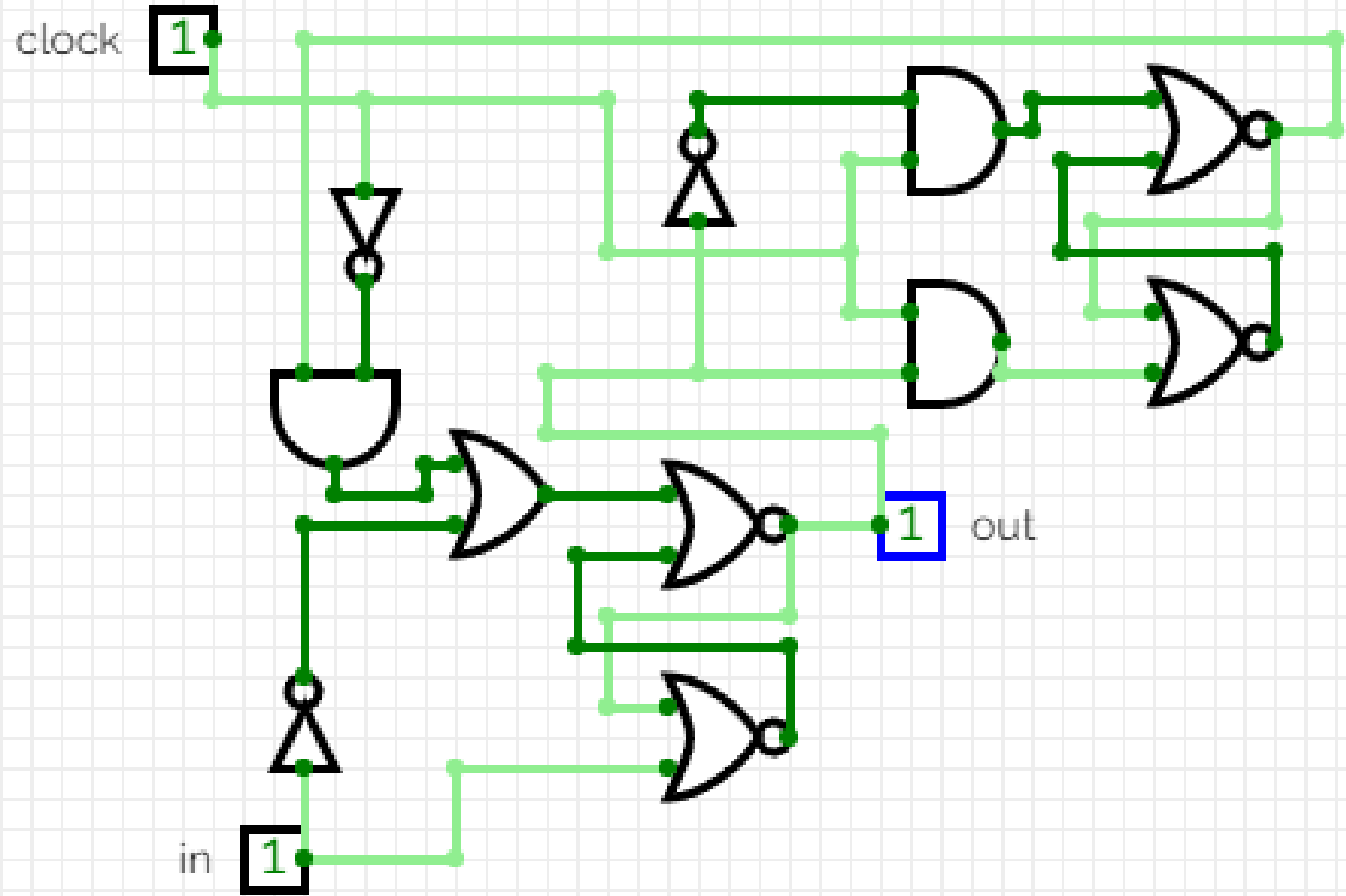
This sub-circuit holds the pre-fixed values of '+' and 'x' in order to output the appropriate operation sign in the 16-segment display.

Registers

Based on the 1-bit register that we learned about in class, we created 4-bit and 8-bit register circuits. These are used to store the values that are inputted into our circuit, as well as the results of operations.



Signal Delay

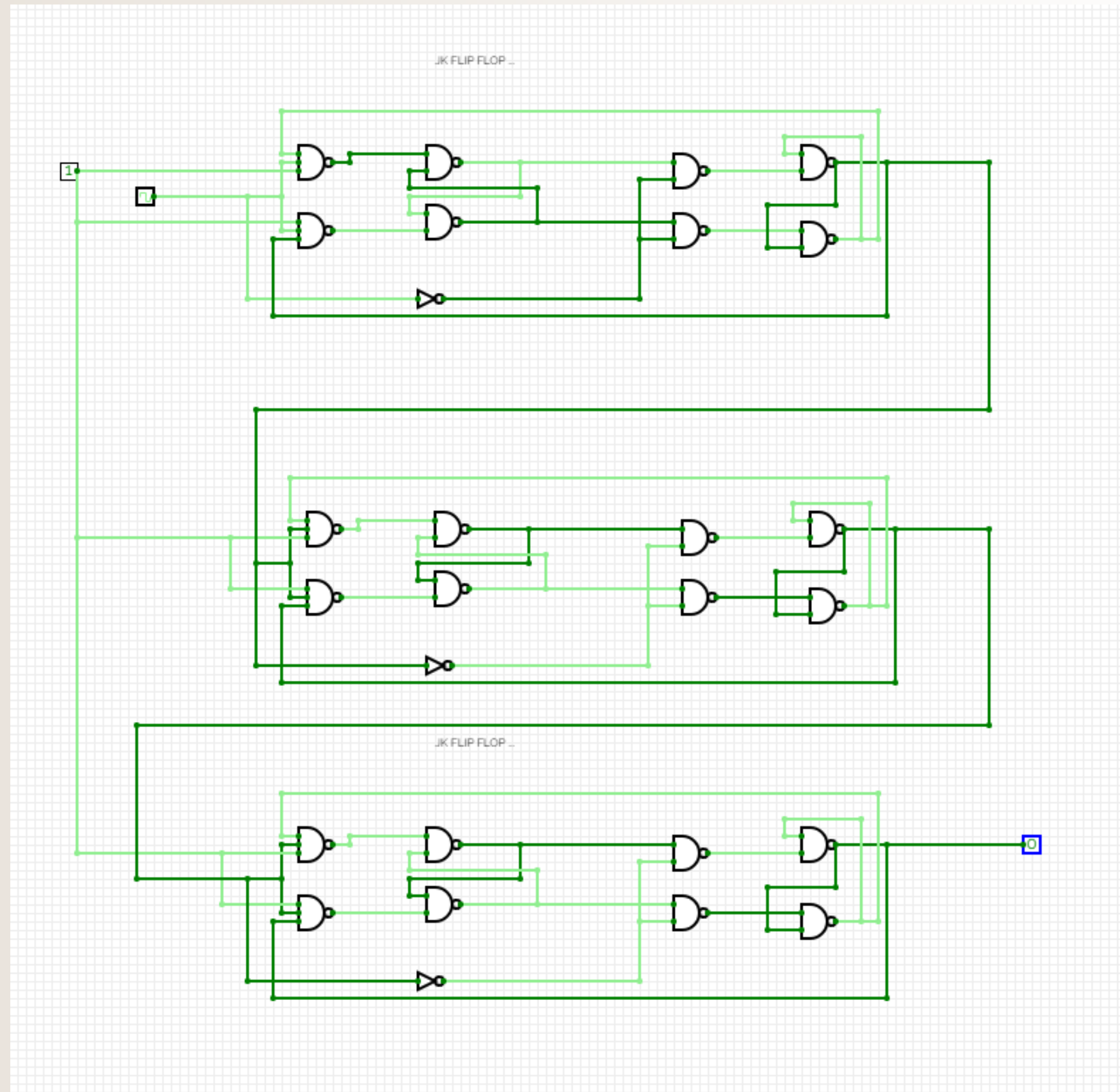


In our effort to mimic real calculators, we created a sub-circuit which re-enters the same input when the user holds the signal button.

In order to achieve this, we needed to figure out a way to interrupt the signal for a short time.

So we came up with the idea of using the clock intervals, which interrupt the user's signal after the initial data passes through the circuit.

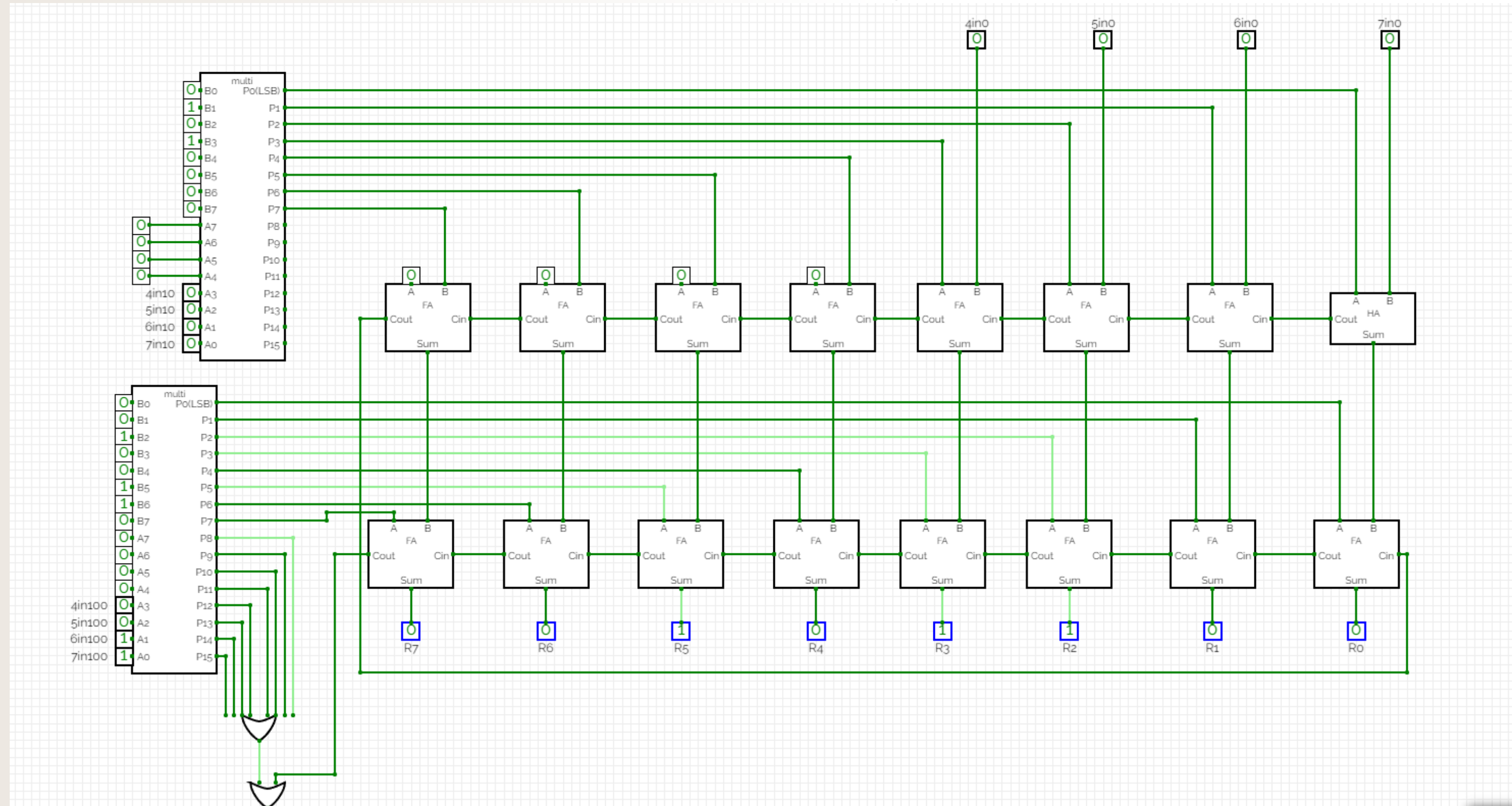
JK Slowing Clock



This is used to slow the clock time - every iteration makes the circuit clock run at half the speed, thus doubling the time between each pulse. We used JK flip-flops to make this possible.

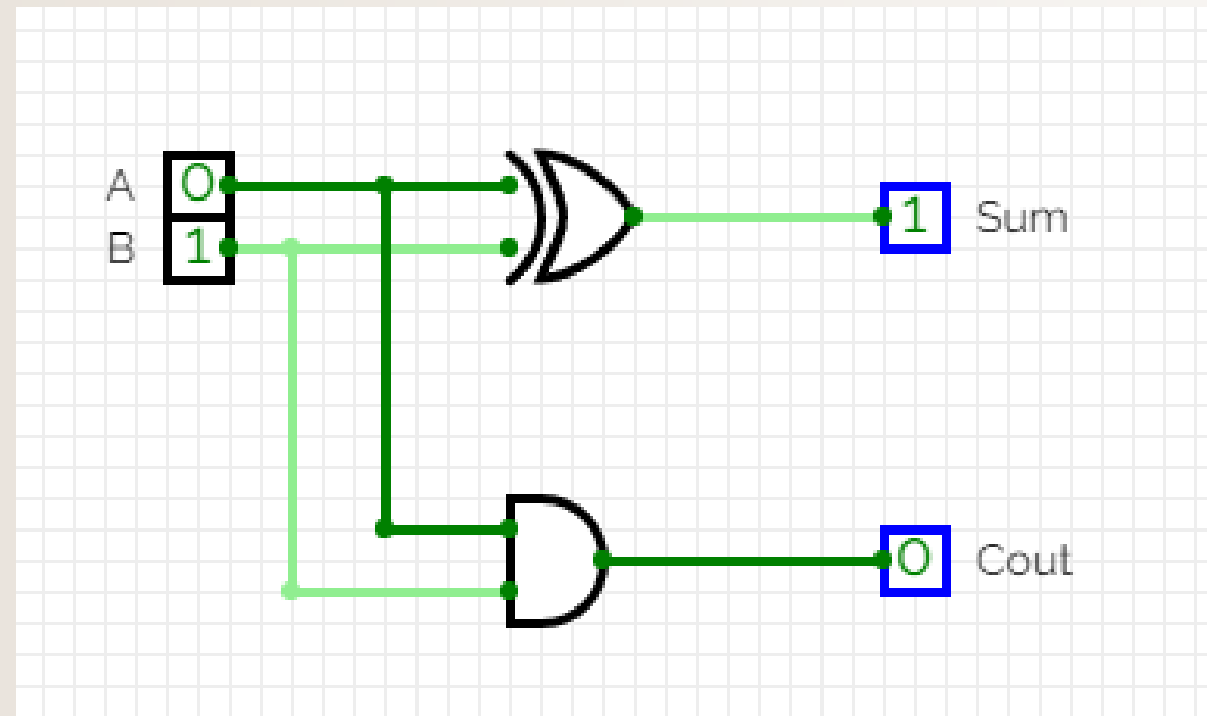
In this way, we are able to use the same clock many times in our circuit, while still having control over the clock speed.

BCD to Binary



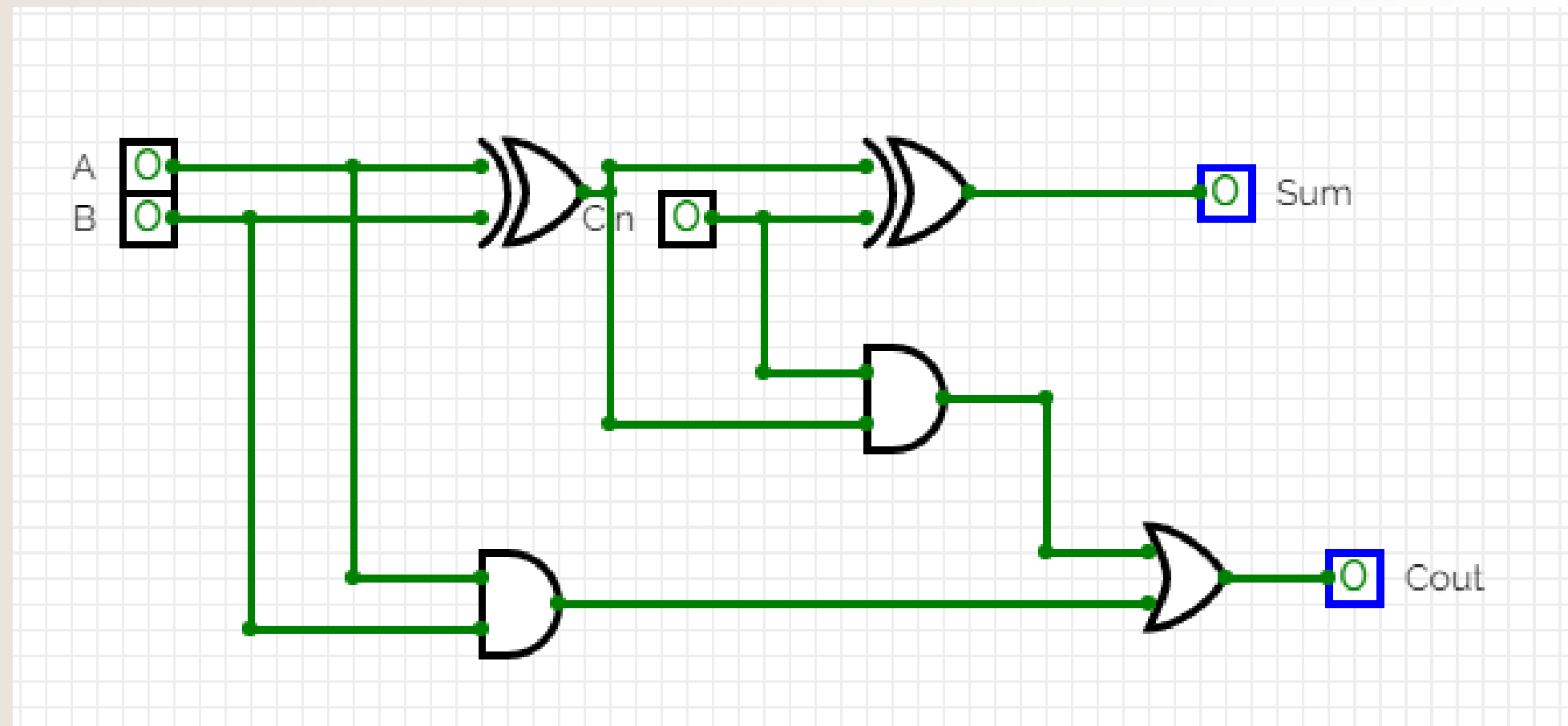
In order to perform the operations, we needed to convert the three numbers of input to one 8-bit number. This sub-circuit accepts up to three 4-bit numbers (0-9), one for the 100s, one for the 10s, one for the 1s. With the help of multipliers and adders, we multiply the first number by 100, the middle number by 10, and then we add them all for a singular 8-bit output.

Half Adder



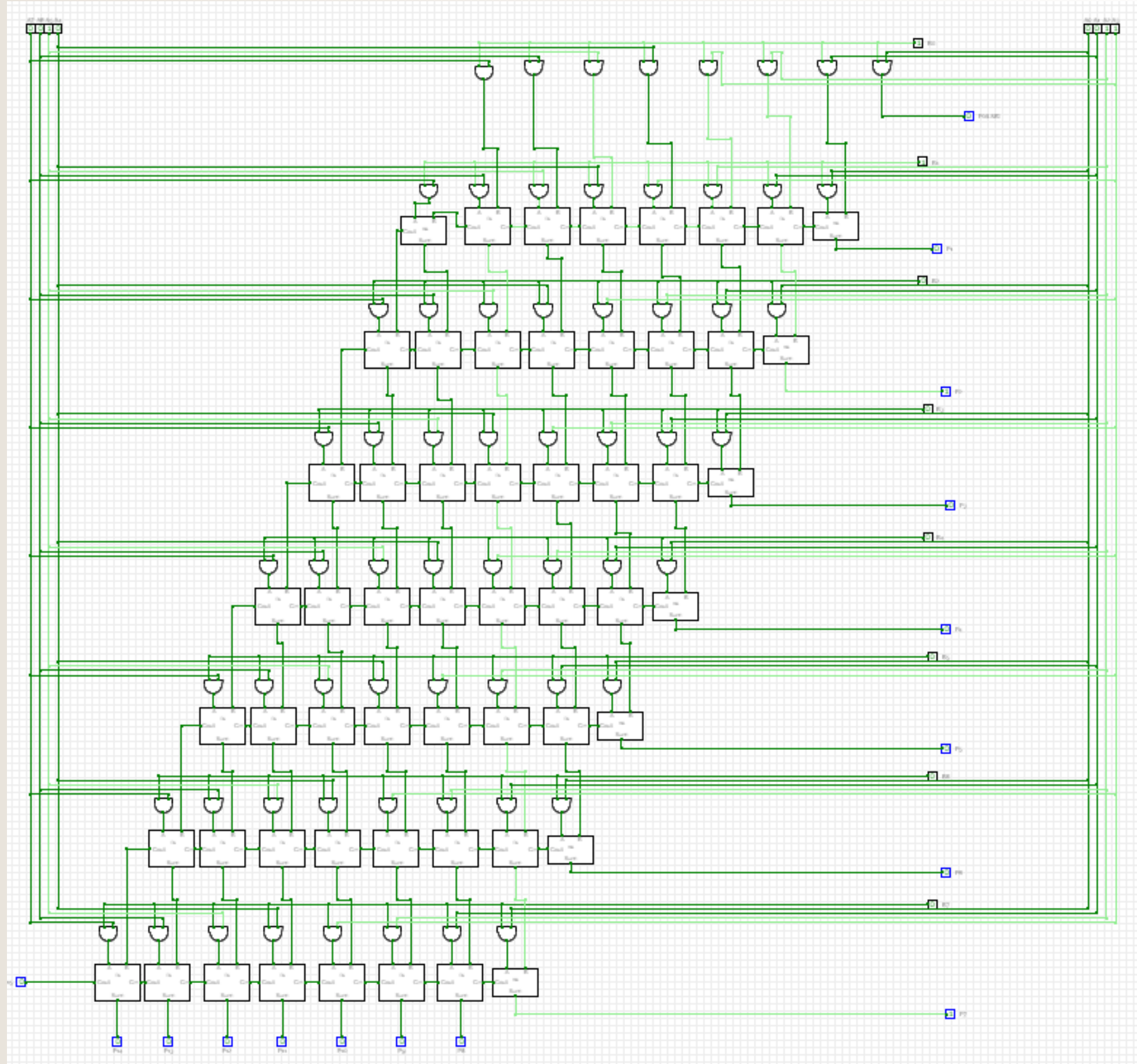
A half adder accepts two bits as input and outputs their sum, as well as a carry output (if present).

Full Adder



A full adder accepts two input bits as well as a carry bit, and outputs their sum, as well as a carry output (if present).

Multiplier

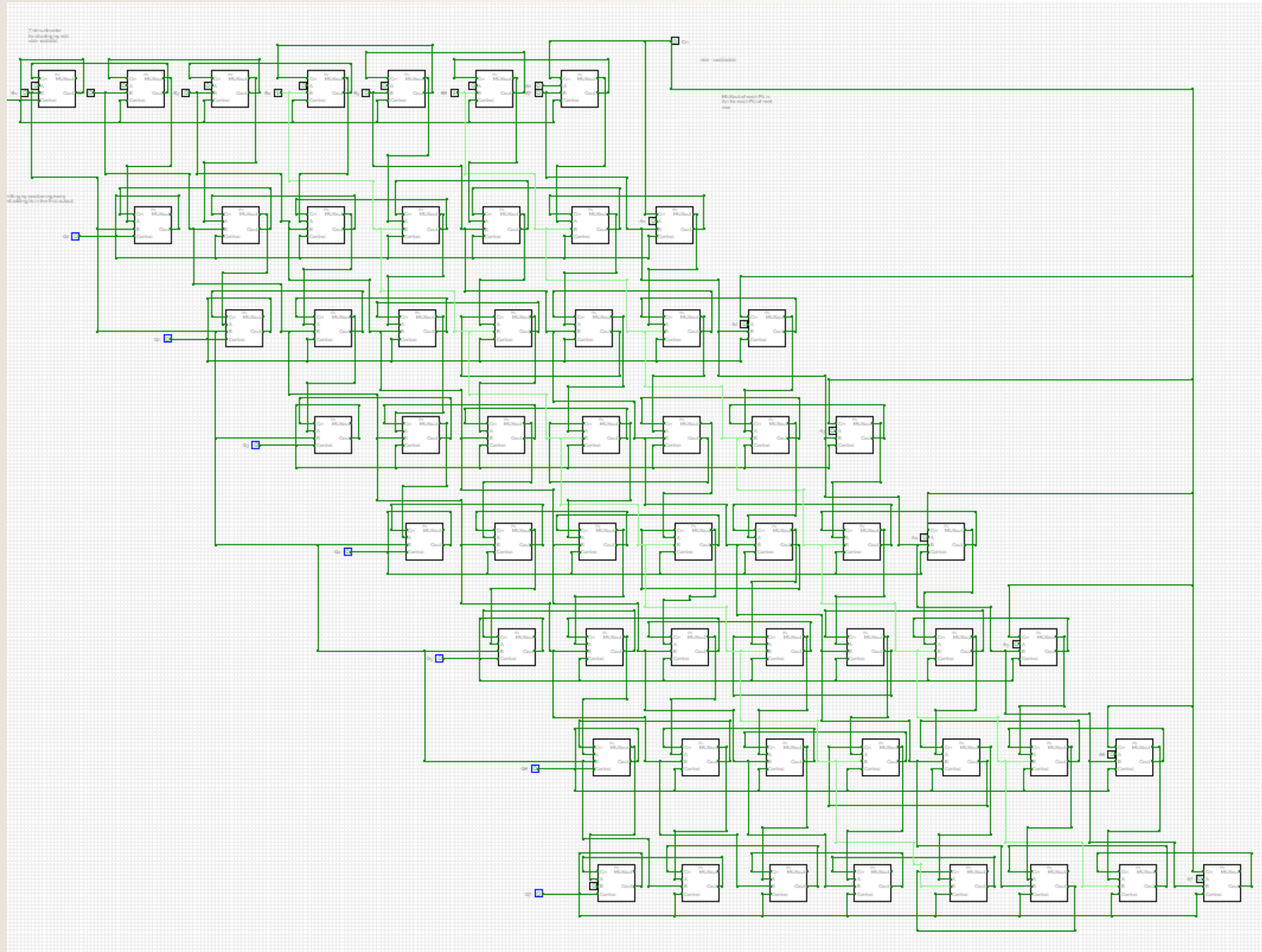


This sub-circuit performs binary multiplication of up to two 8-bit numbers.

We used a combination of AND gates, half adders and full adders, making sure to left shift by one at the end of each iteration.

The multiplier outputs 16 bits: the 8 most significant bits are inputted into an OR gate, the output of which serves as an error indicator.

Divider

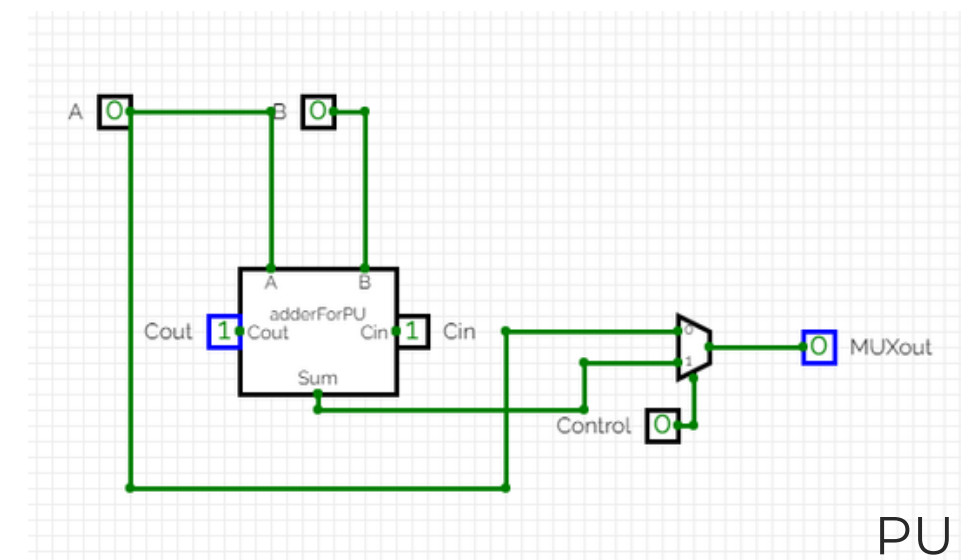


This sub-circuit performs binary division of up to two 8-bit numbers.

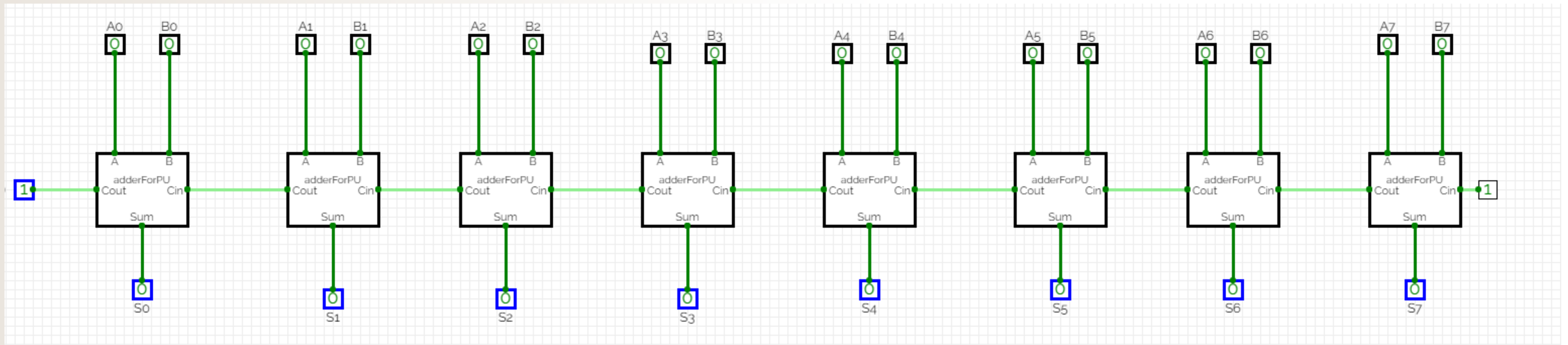
The bits are inputted into several processing units built out of one full adder and one multiplexer.

The control signal of the multiplexers (in each row) is determined by the carry output of the last adder of the row.

At the end of each iteration, we right shift by one and keep the quotient bits as output.

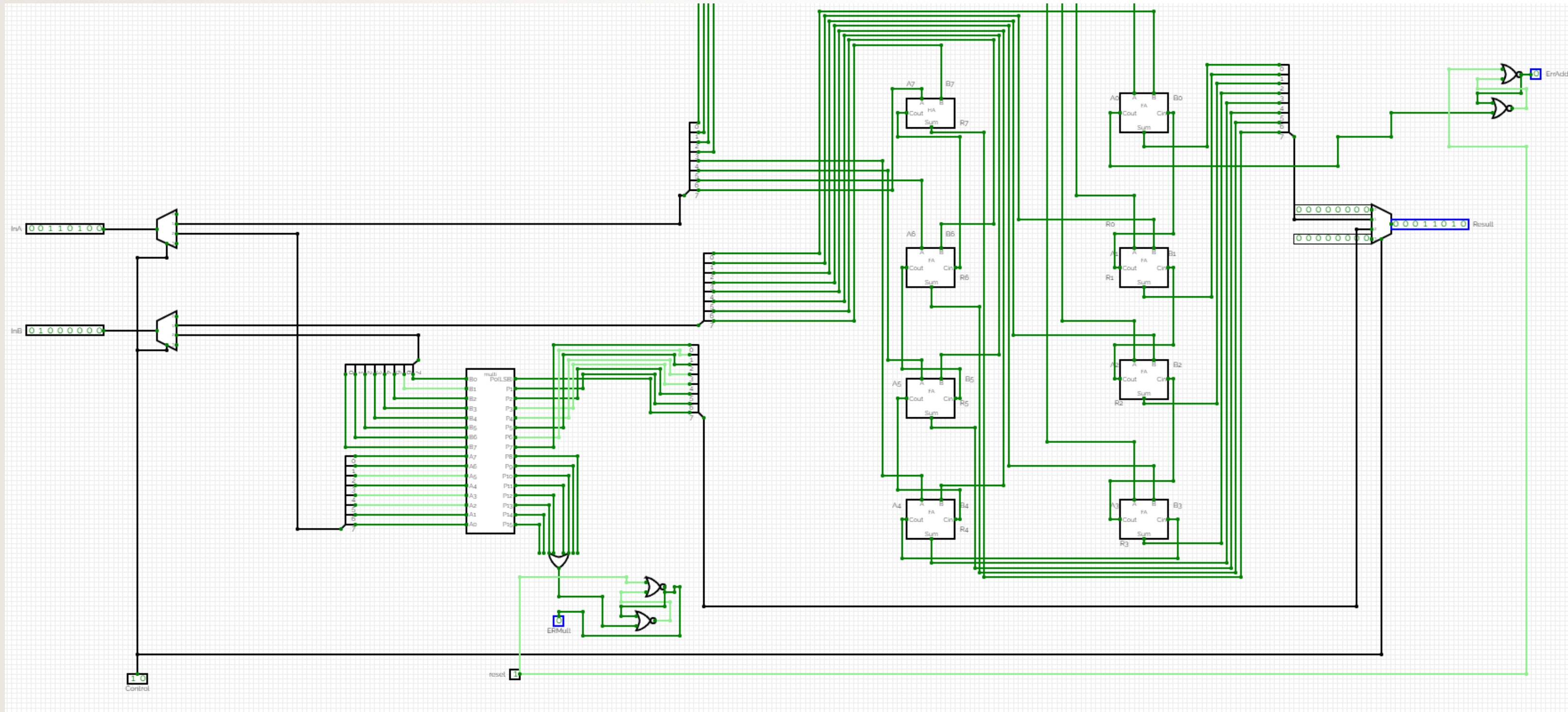


Subtractor



This sub-circuit performs binary subtraction. It accepts two 8-bit numbers as input, converts the second number into a 2's complement of that number and performs the subtraction.

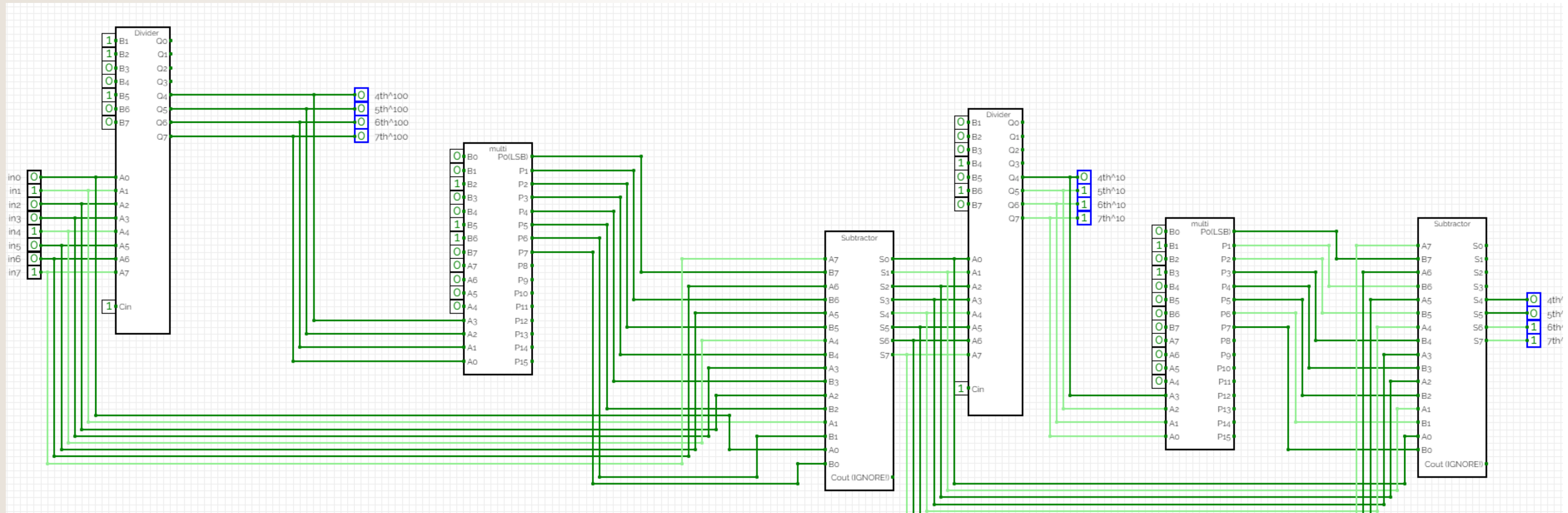
Arithmetic Logic Unit



Accepts two 8-bit numbers, as well as a control signal that decides the operation (addition or multiplication).

Outputs the result of that operation or an error indicator.

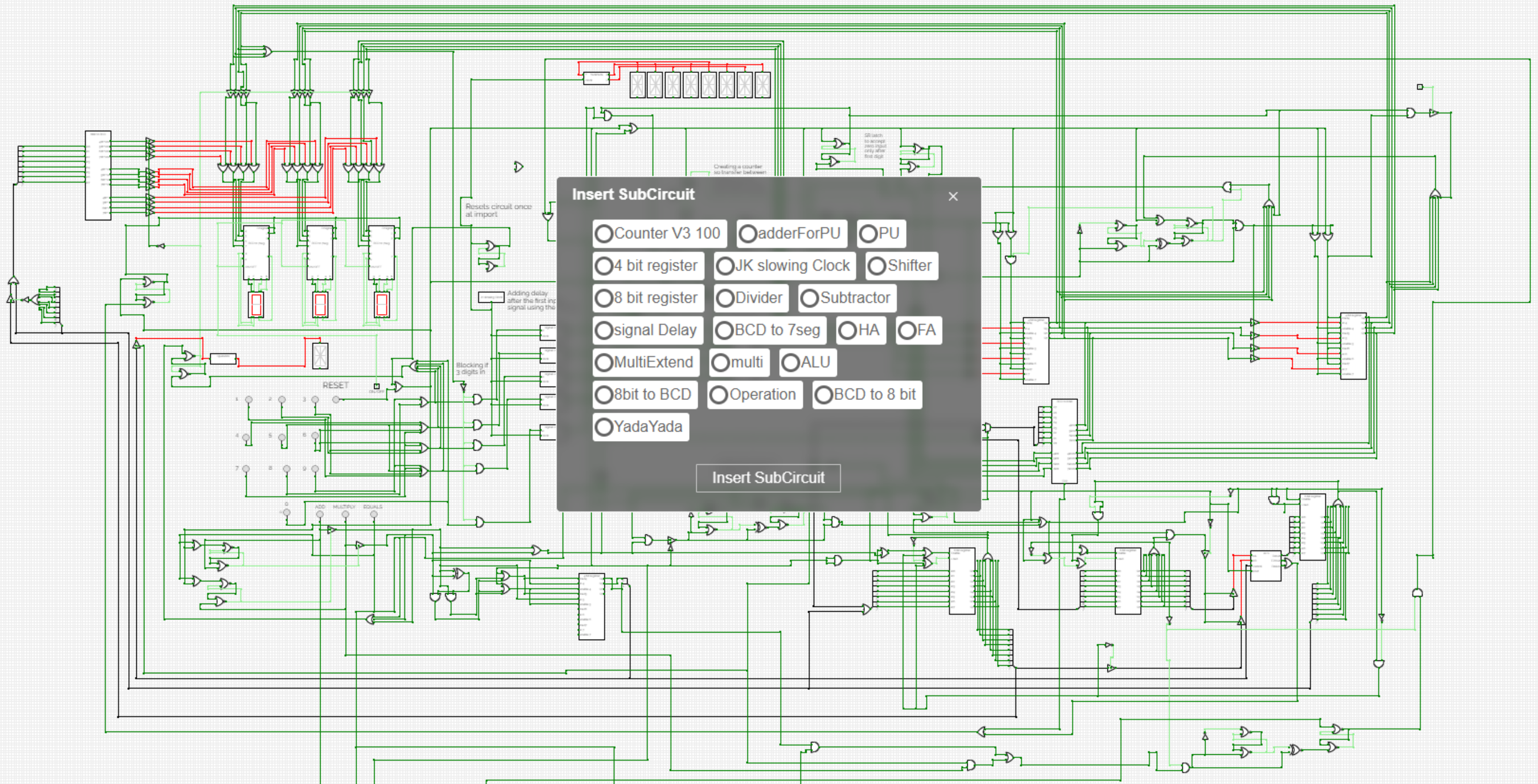
Binary to BCD Decoder



We then needed to perform one final operation, in order to convert the output of the ALU into three numbers, which would then be displayed to the user. For this purpose, we perform the opposite operations of the 'BCD to binary' sub-circuit; we apply a combination of division, multiplication and subtraction.

This outputs one number for the 100s digit, one for the 10s, and one for the 1s.


OUR FINISHED PROJECT




CONTACT US

The Team:

Balasis Ioannis
Kokkinos Georgios
Kouridaki Anastasia
Ververaki Argyro

 YadaYadaAT

 yadayadaAt@gmail.com

