AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING
I-CREDIT HOURS ENGINEERING PROGRAMS COMMUNICATION SYSTEM ENGINEERING

CSE222: Digital Logic Design

Arithmetic Logic Unit

Summited to:

DR. Hossam Hassan
ENG. Dina Zakaria

Submitted by:

1) Ahmed Belal 23P0007
2) Omar Ahmed 23P0369
3) Naira Ahmed 23P0408
4) Rawan Waleed 23P0119
5) May Walid 2200259

Table of Contents

1.0 Introduction

This project focuses on designing, simulating, and implementing a digital system to perform arithmetic and logical operations with output displayed on a seven-segment display. The system features a 4-bit Arithmetic Logic Unit (ALU) capable of executing signed addition, subtraction, multiplication, and division operations using two signed 4-bit operands.

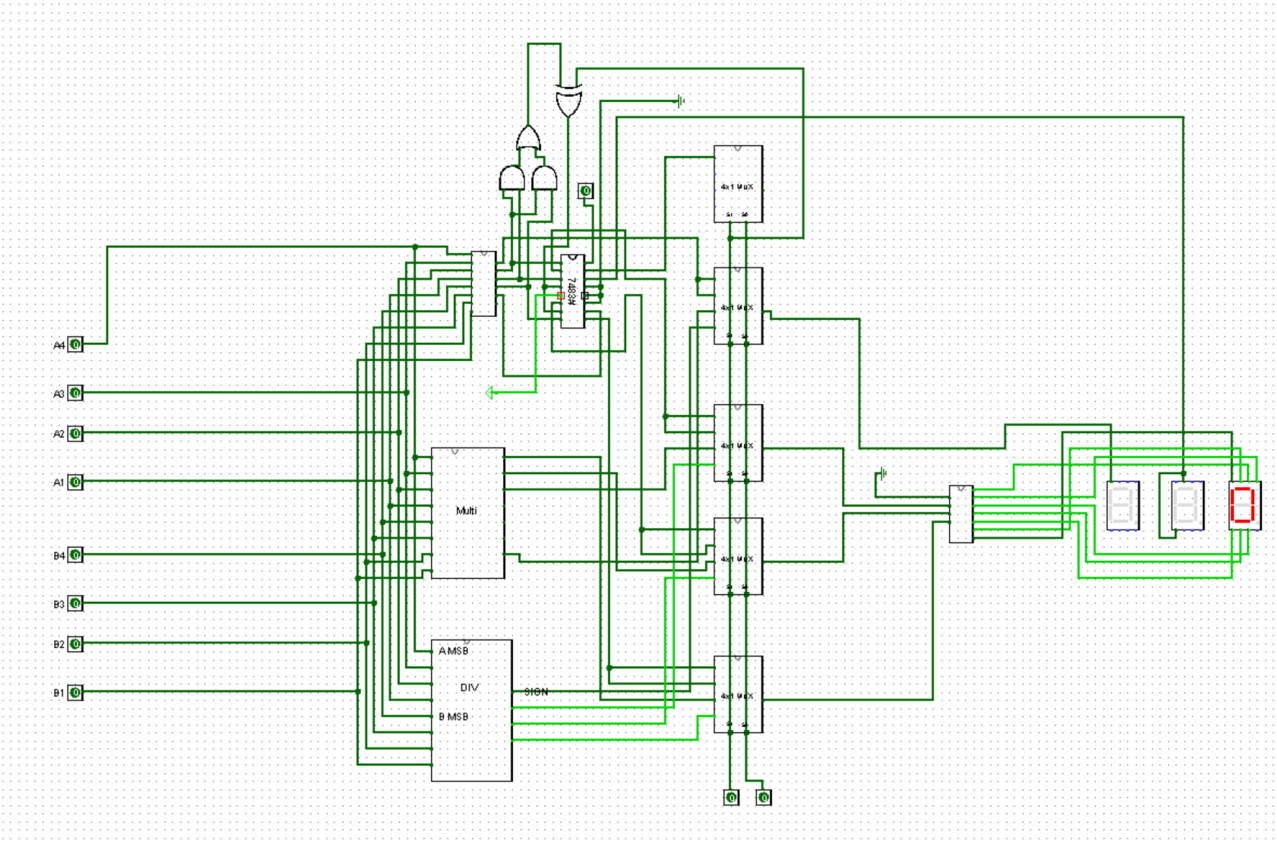
For addition and subtraction, we implemented two output handling modes. In the first mode, results range from -7 to +7, accompanied by an overflow indicator LED that lights up when the result exceeds this range. In the second mode, the full signed range from -14 to +14 is supported without triggering overflow, offering greater accuracy and flexibility in representation.

Multiplication is performed as a 3-bit × 3-bit signed operation, and the resulting 6-bit output ranges from -7 to +7 when shown on the schematic's seven-segment display, due to visual space limitations. However, to demonstrate the full output range (0 to 49), we displayed the complete 6-bit result on LEDs. Additionally, we designed a separate circuit that converts the 6-bit binary multiplication result into two-digit BCD format (8-bit total) so that it can be shown accurately on two seven-segment displays. This BCD conversion circuit was implemented and tested on a virtual simulator but was not integrated into the main schematic.

This phase of the project emphasizes the simulation of each arithmetic operation, validation of results, and ensuring that the system correctly handles overflow, signed operations, and output representation. Through this work, students deepen their understanding of digital logic design, binary arithmetic, and hardware-level interfacing output.

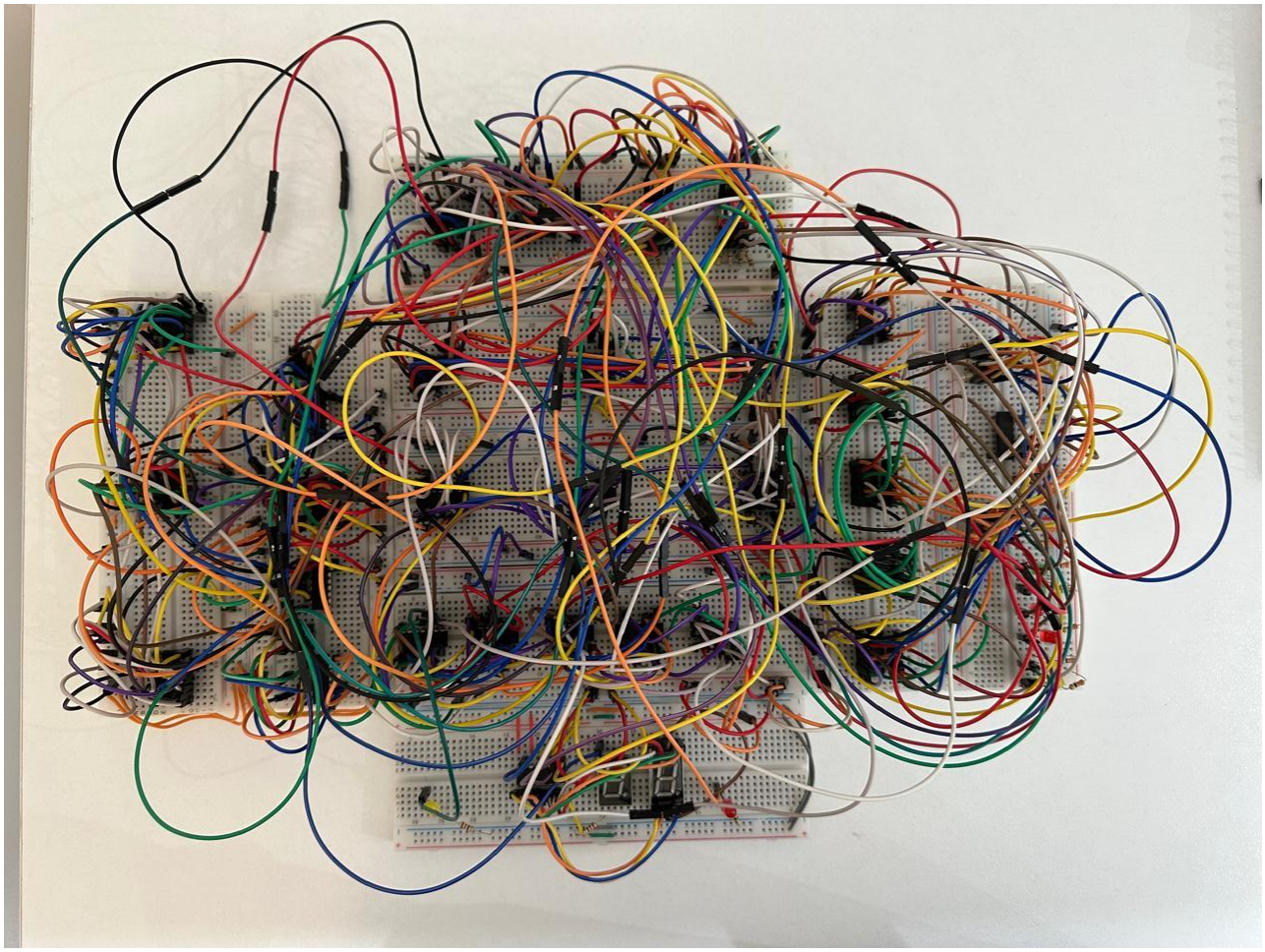
2.0 ALU Circuit Simulation

- Displaying Addition & Subtraction from -14 to 14
- Displaying Multiplication from -7 to 7



3.0 ALU Implementation Hardware

- Displaying Addition & Subtraction from -7 to 7
- Displaying Multiplication from -7 to 7
- Displaying Multiplication from -49 to 49 on LEDS



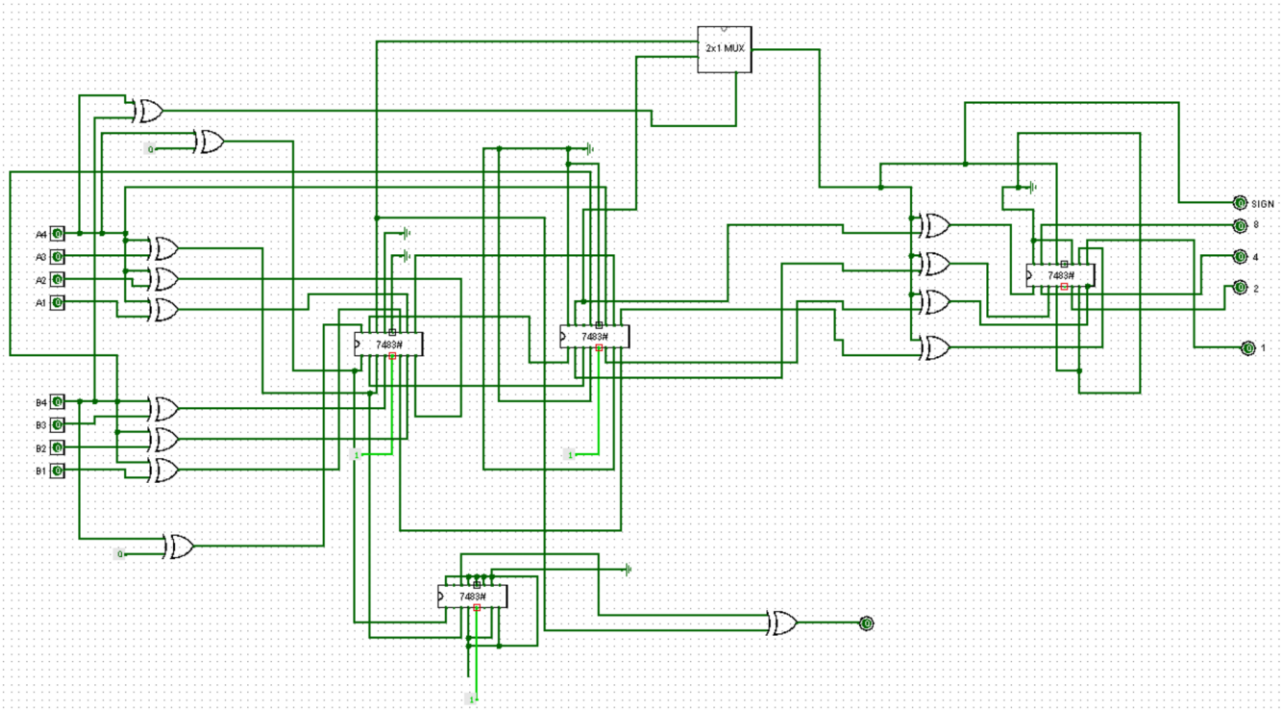
4.0 Addition & Subtraction:

For the addition and subtraction operations, we implemented a general-case circuit that supports both functions using a shared arithmetic unit. The operation is selected using a 2-bit control signal: when the control input is set to 00, the circuit performs addition; when set to 01, it performs subtraction. This selection is managed through a multiplexer that directs the correct operands into the adder based on the selected mode.

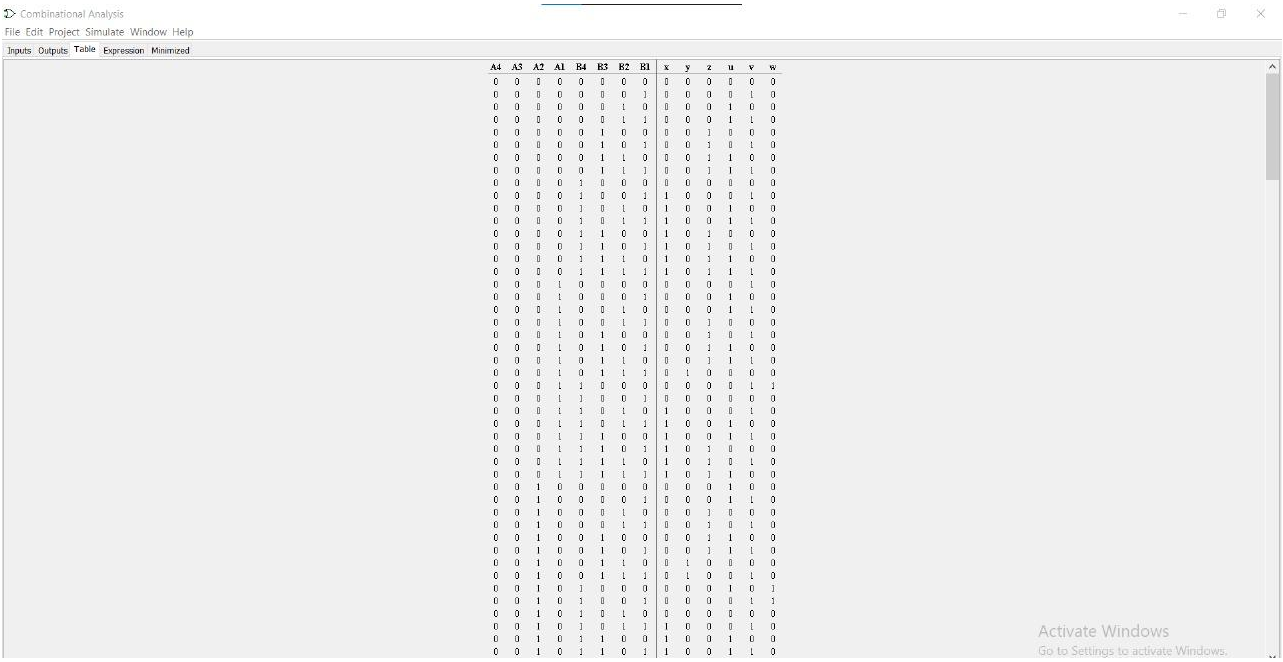
To handle arithmetic overflow, we added an overflow indicator LED in the hardware implementation. This was necessary because the hardware circuit displays results only within the range of -7 to +7. If the result exceeds this range, the overflow LED is activated. To expand the display capability, we designed a separate BCD adder circuit that converts the binary output to BCD format, allowing us to accurately represent values from -14 to +14 using five bits. This BCD circuit is included in the full simulation ALU design, but it was not used in the hardware due to hardware limitations.

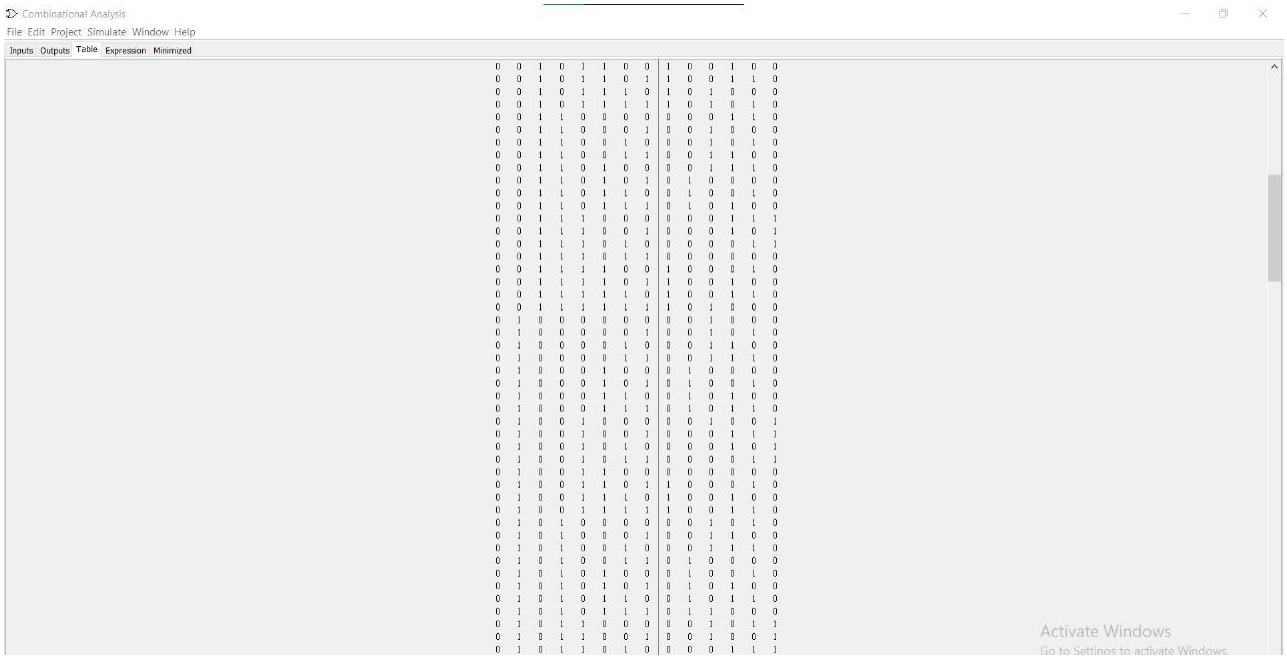
In the hardware setup, we used the limited -7 to +7 circuit along with the overflow LED for practical demonstration on the lab. In contrast, the full ALU simulation includes the BCD adder to show the full result range from -14 to +14 and is used to demonstrate both the correct output and the overflow condition in software. To clarify, we present the -7 to +7 circuit with overflow as a standalone design, while the complete ALU simulation and lab-assembled version include the extended BCD functionality.

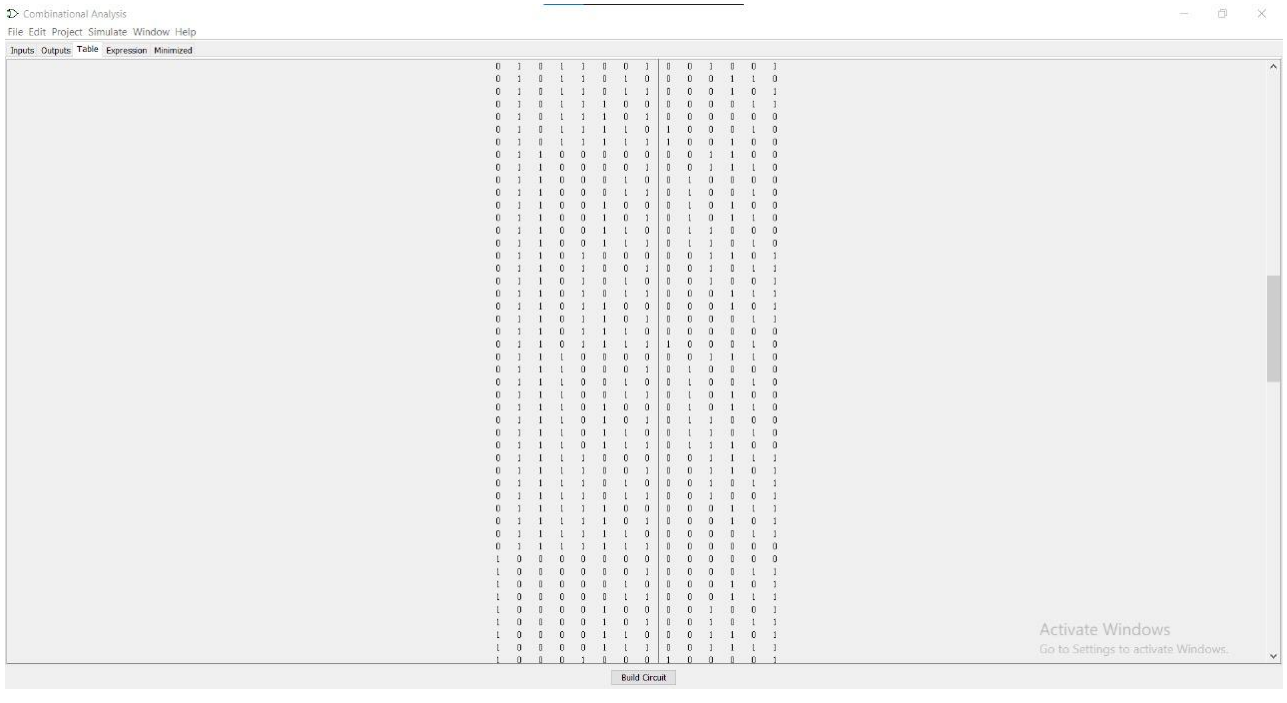
4.1. Addition & Subtraction (-14 to 14):

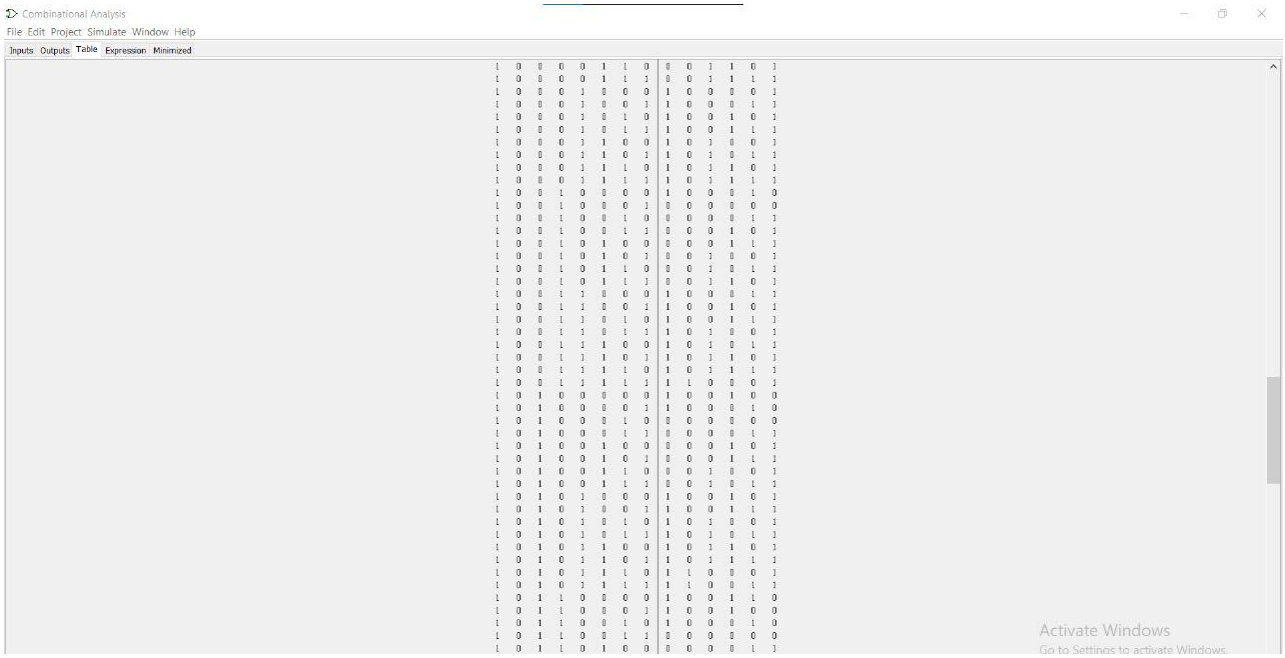


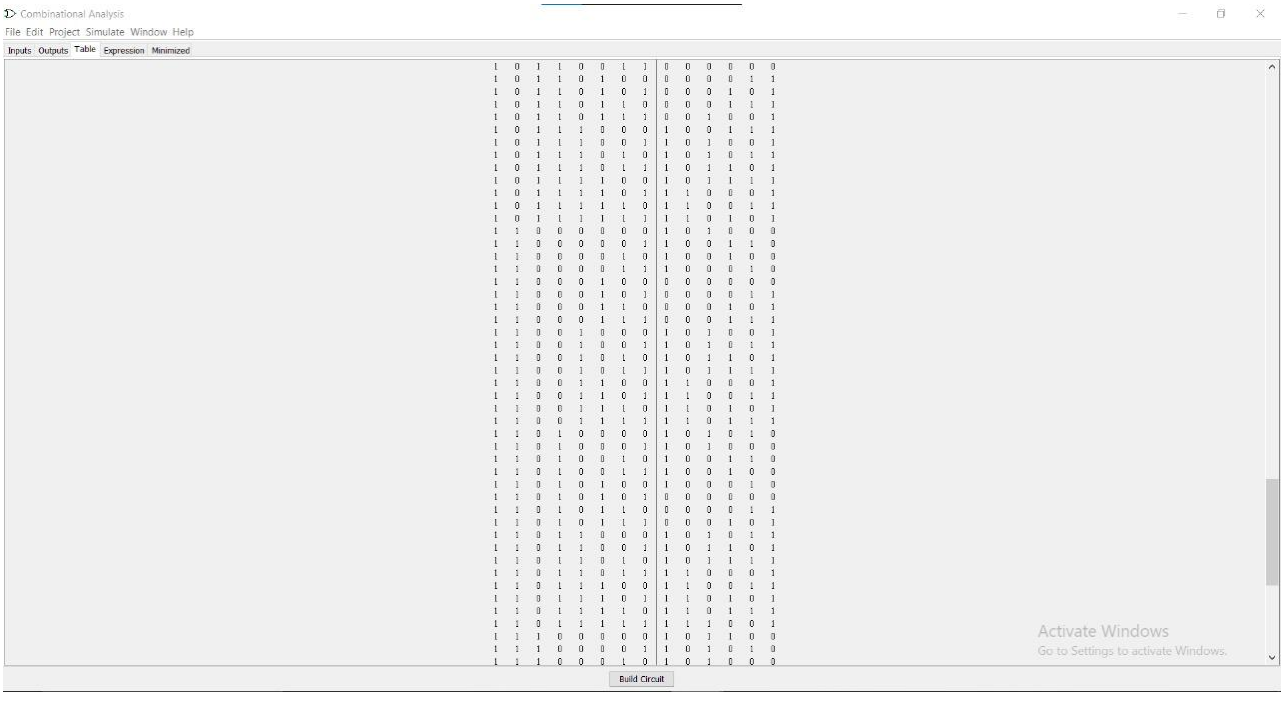
4.2. Truth table (-14 to 14):

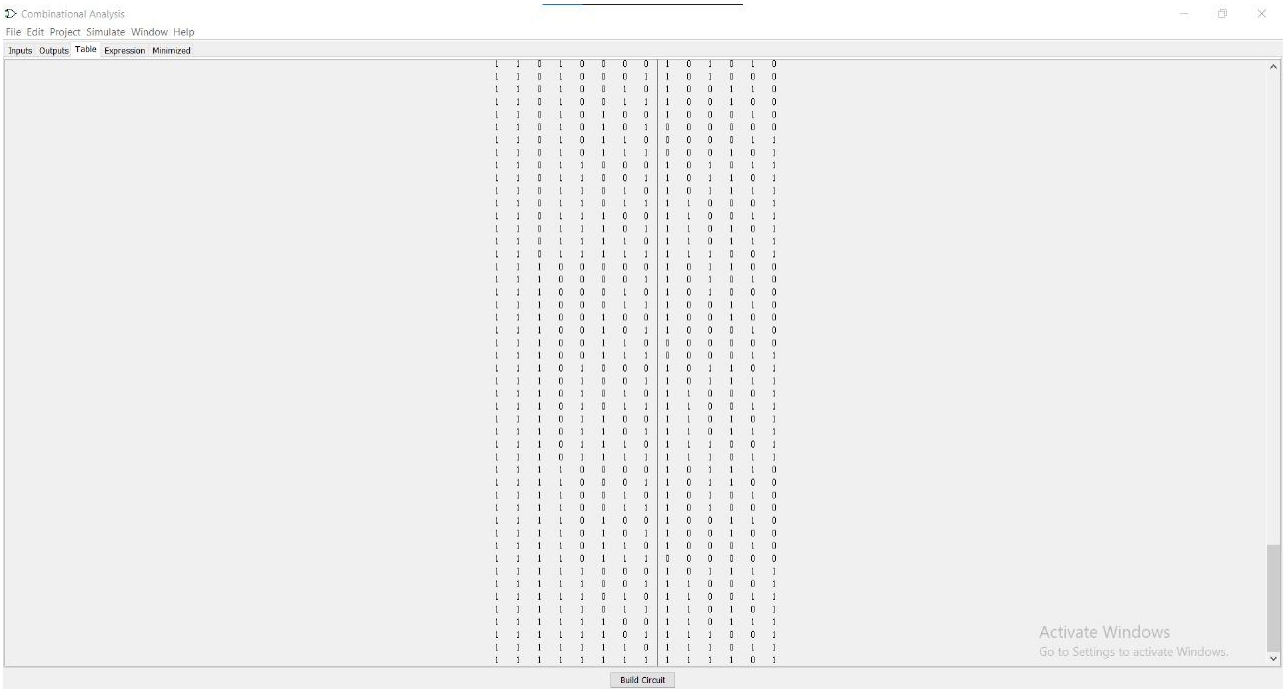



Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.

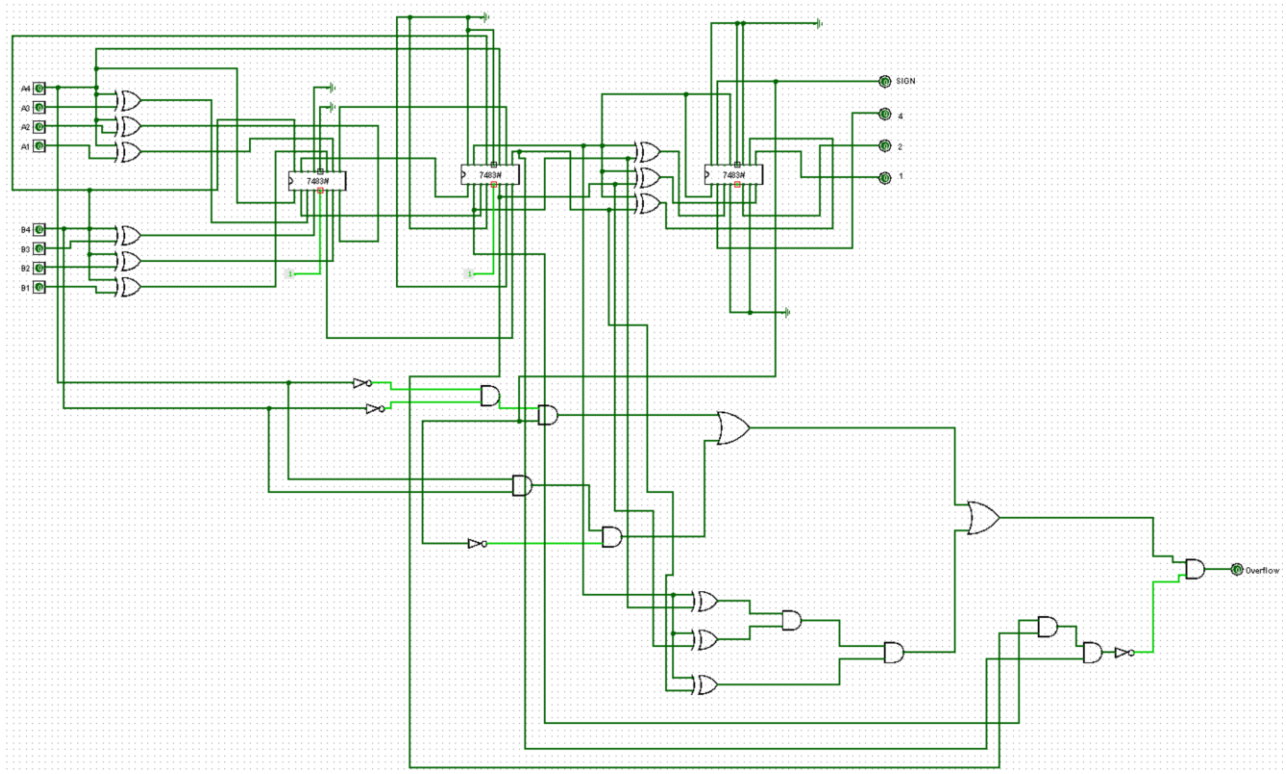

Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.
Build Circuit


Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.

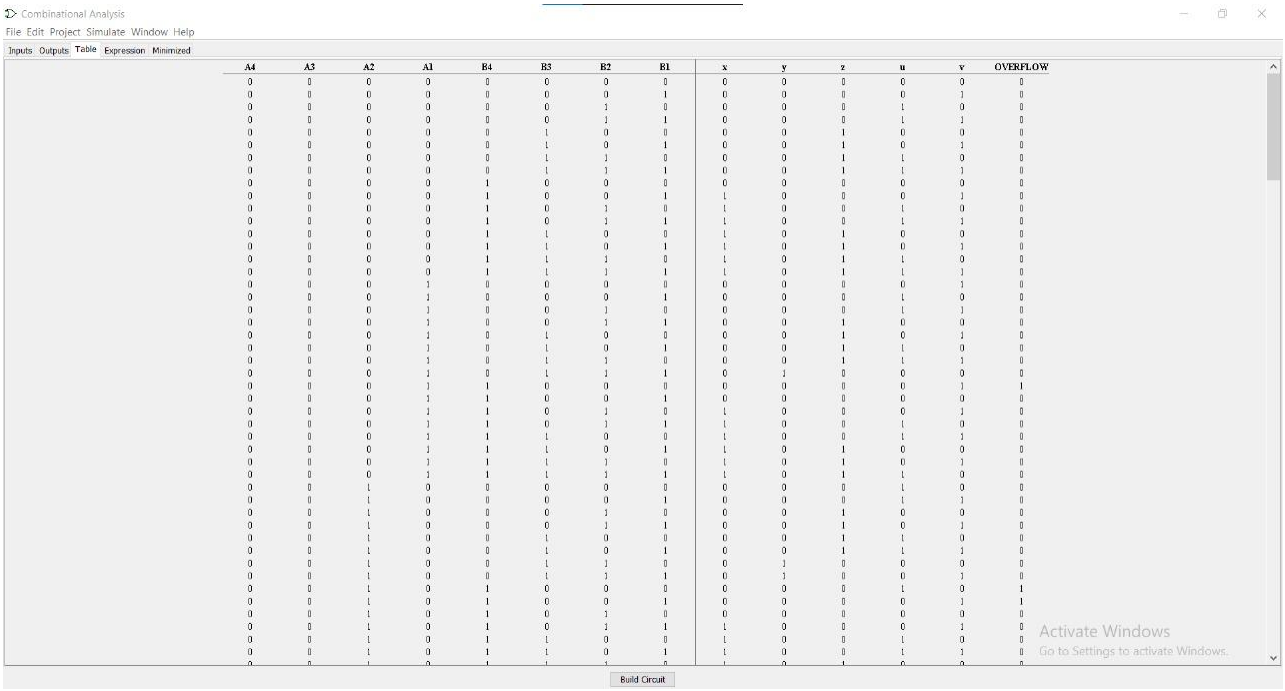

Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.
Build Circuit

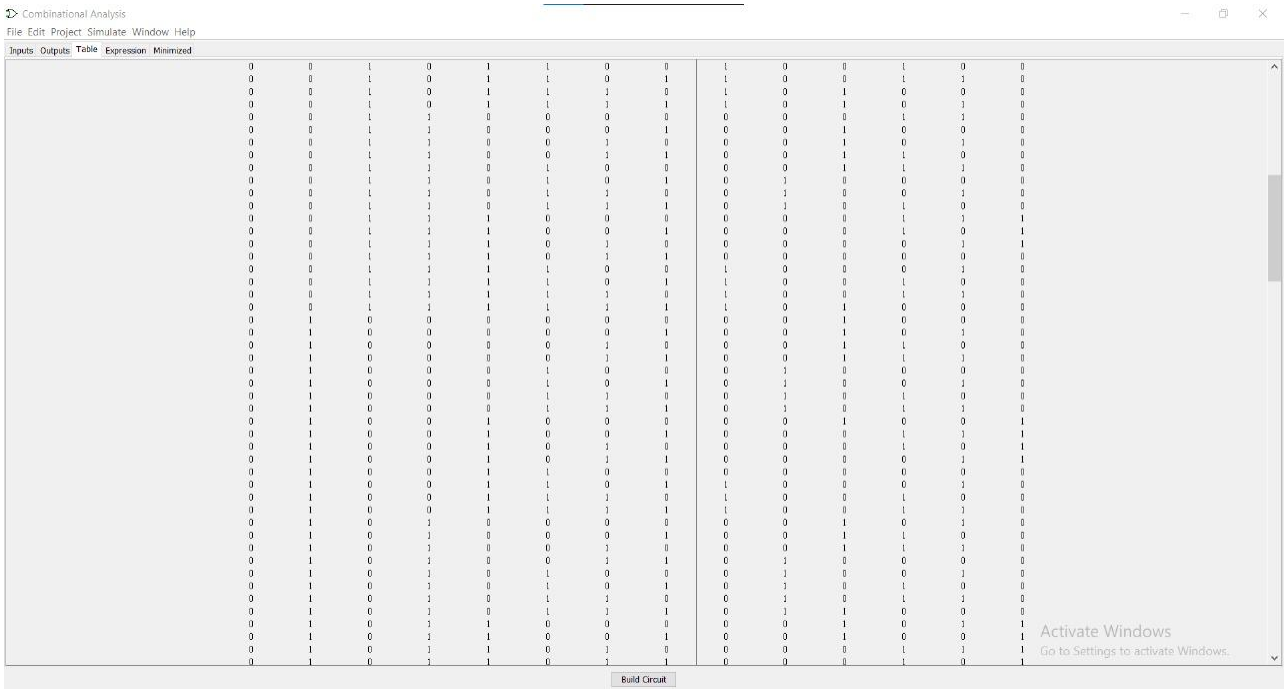


4.3. Addition & Subtraction (-7 to 7 & Overflow):

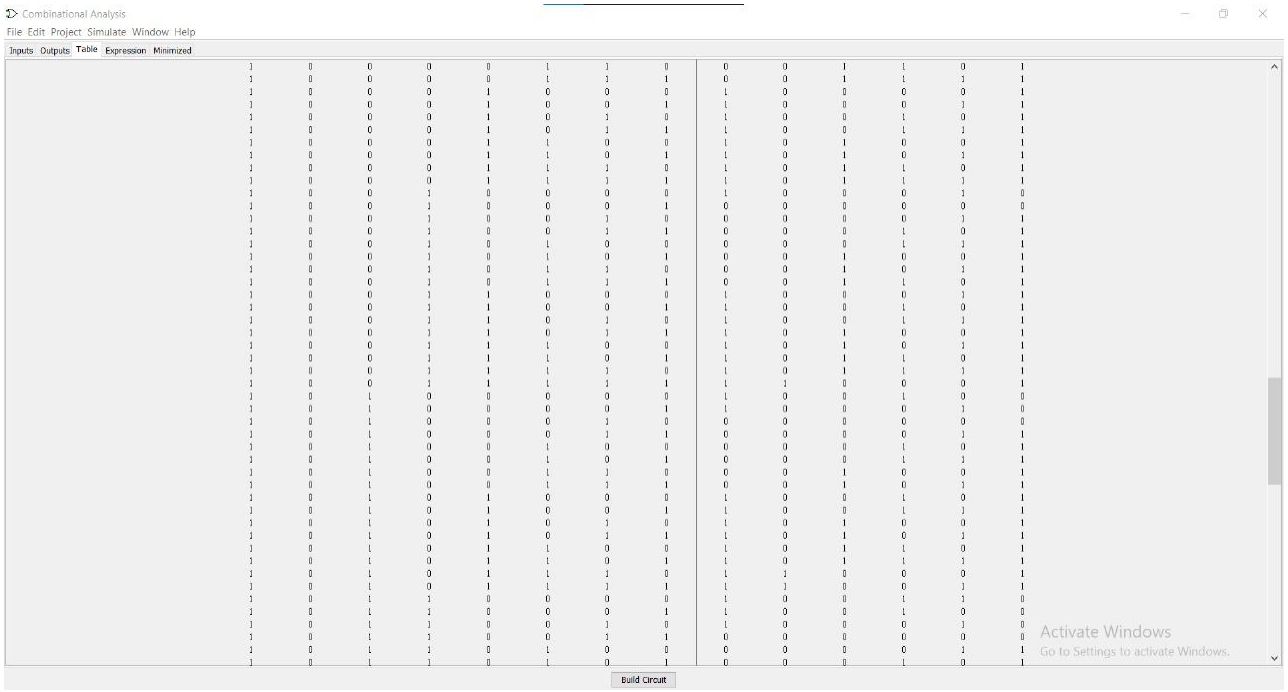


4.4. Truth table (-7 to 7 & Overflow):




Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.
Build Circuit

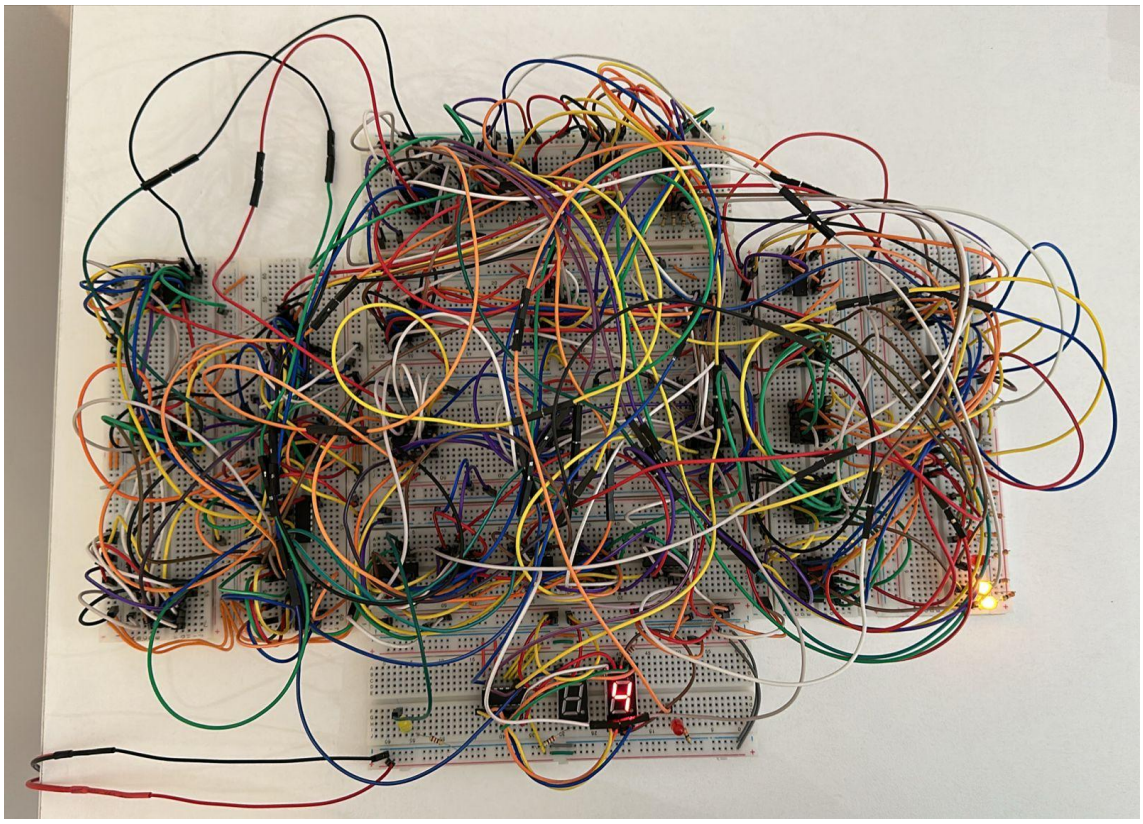

Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.
Build Circuit

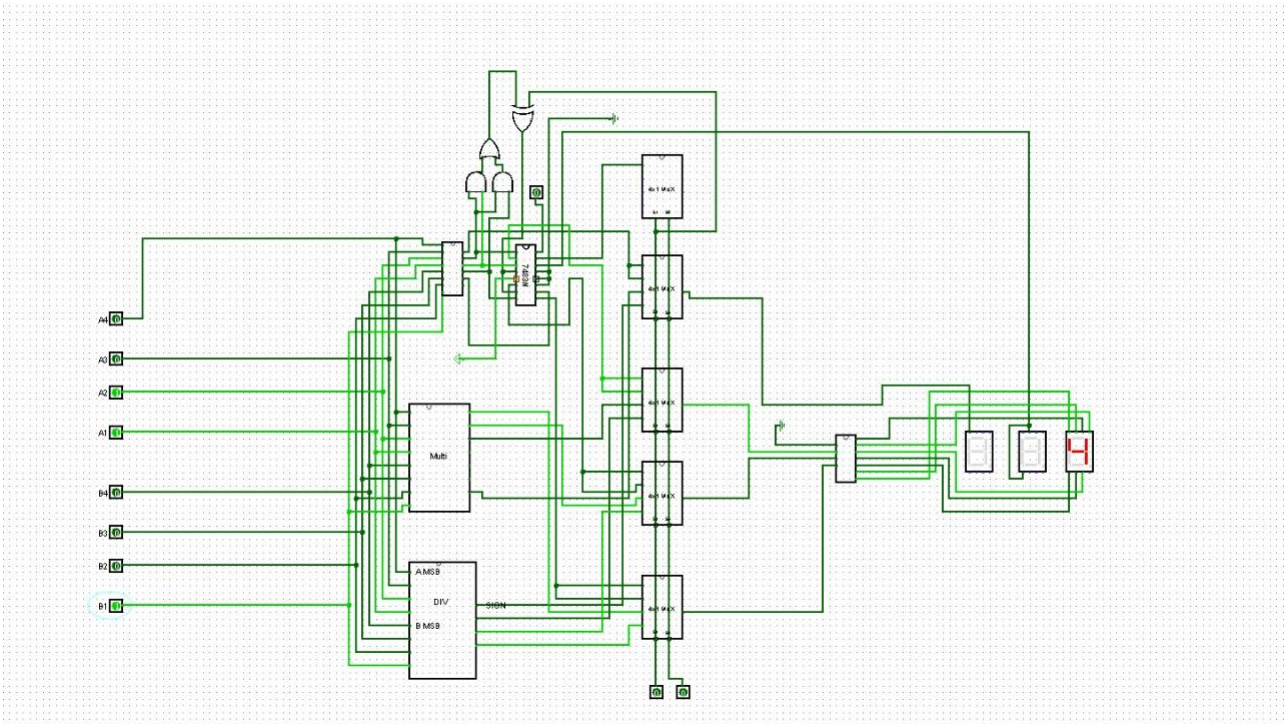

Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.
Build Circuit


Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.
Build Circuit

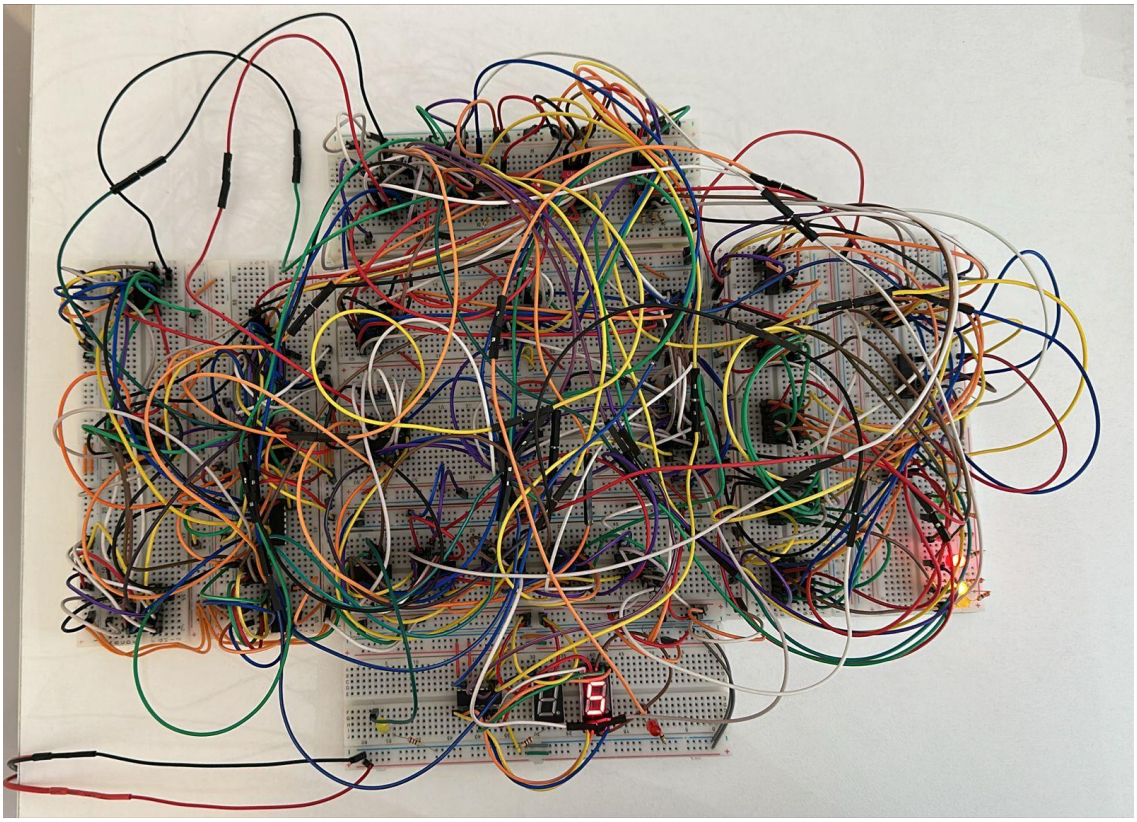
4.5. Cases

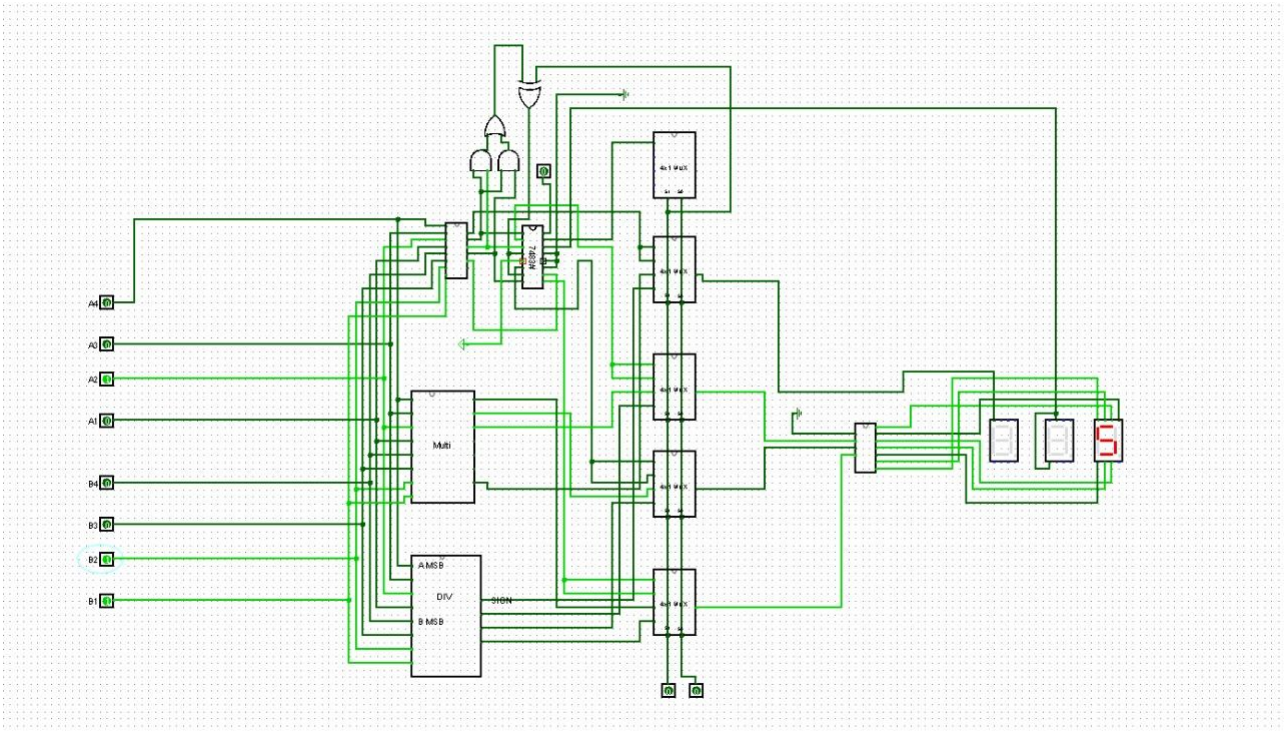
- Case (3+1)



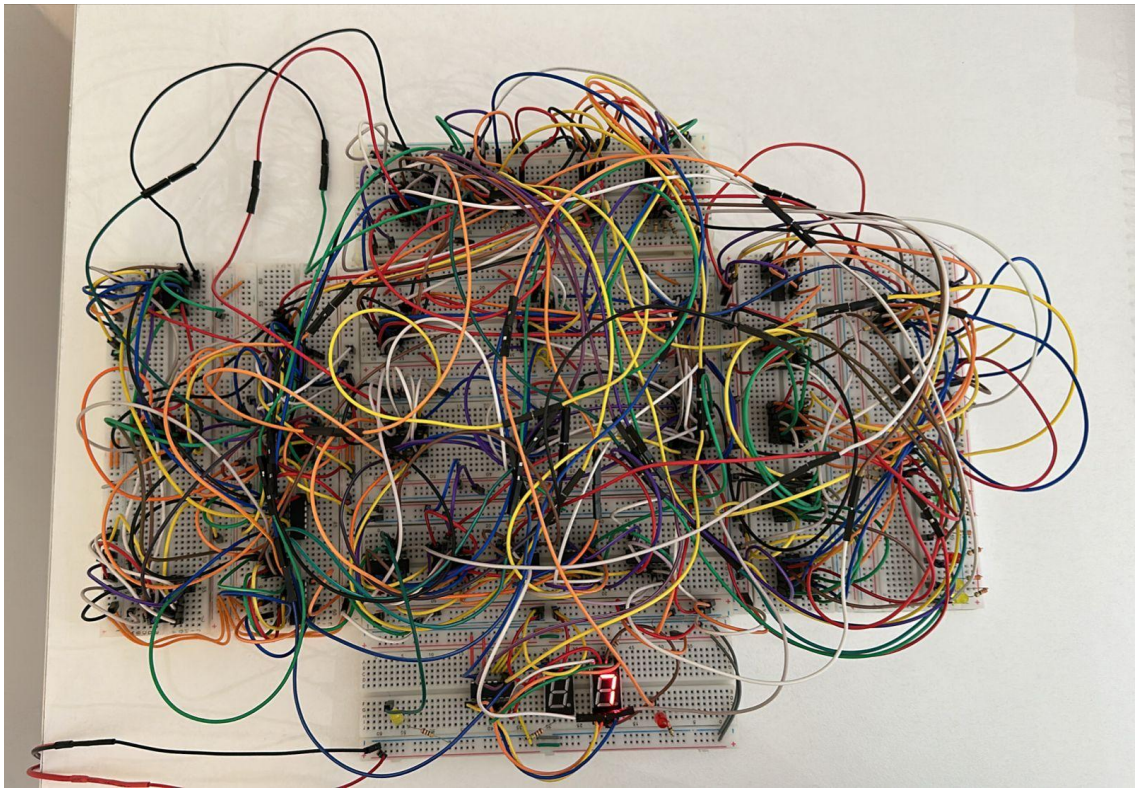


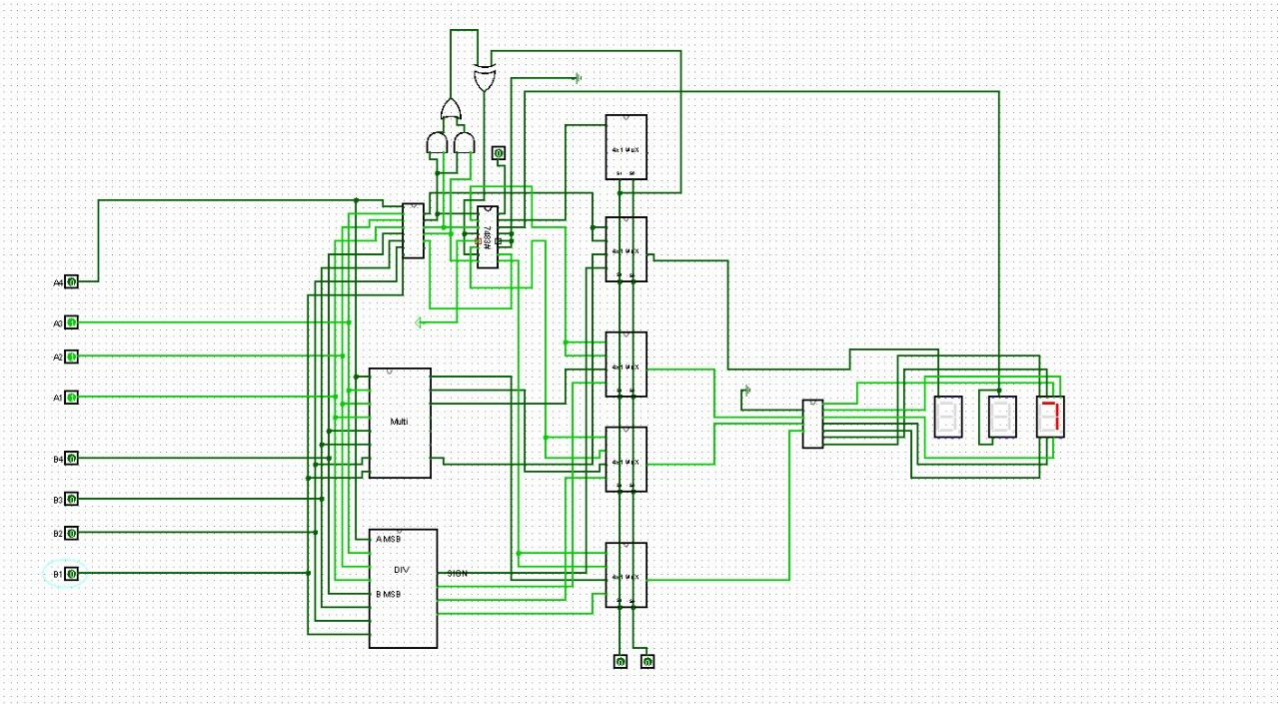
- Case (3+2)



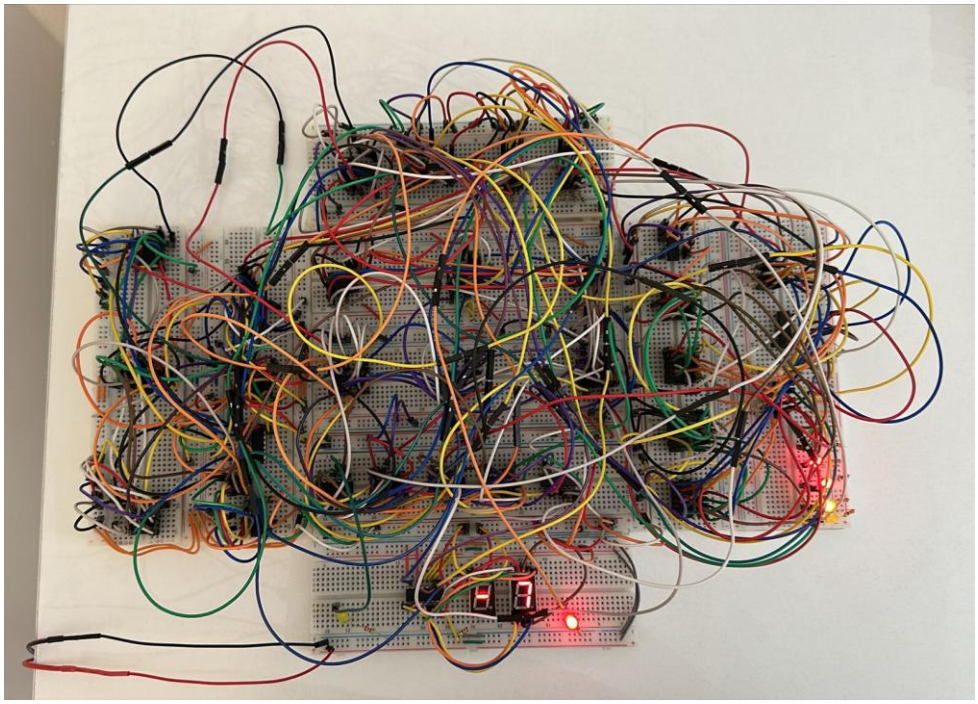


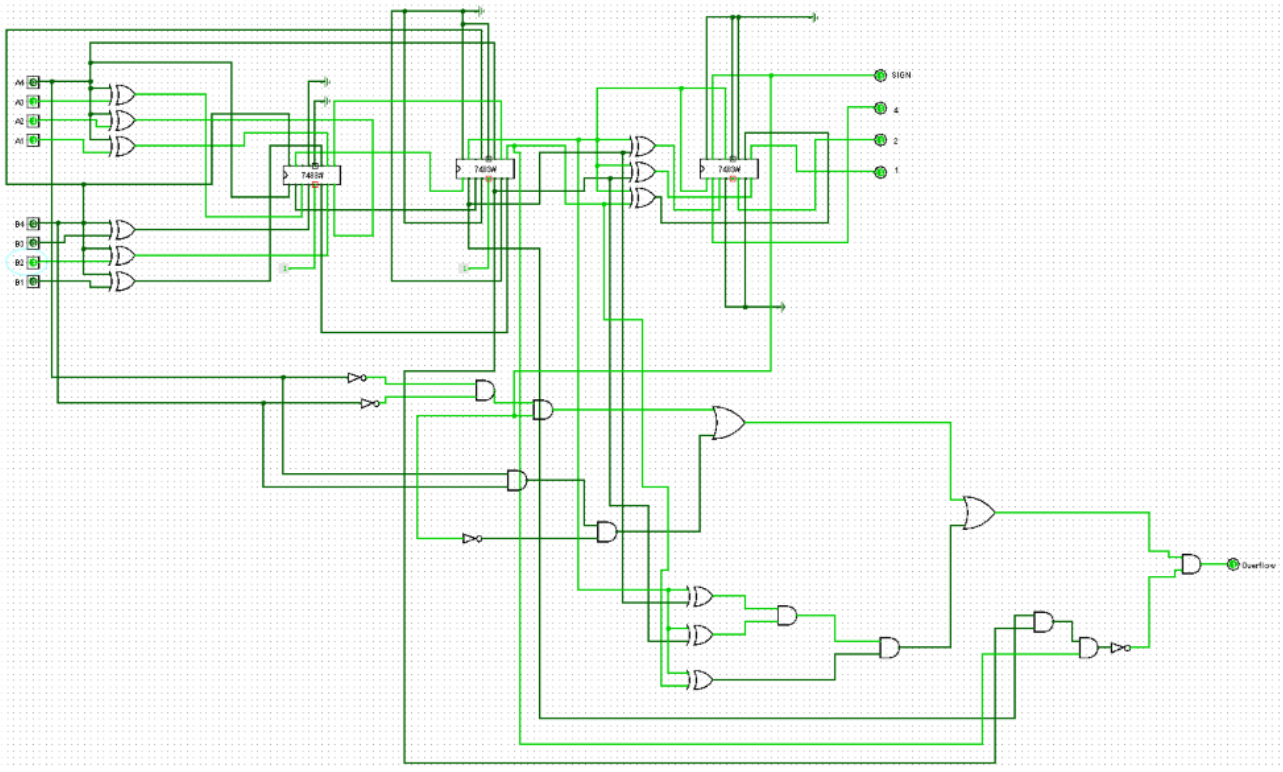
- Case (7+0)



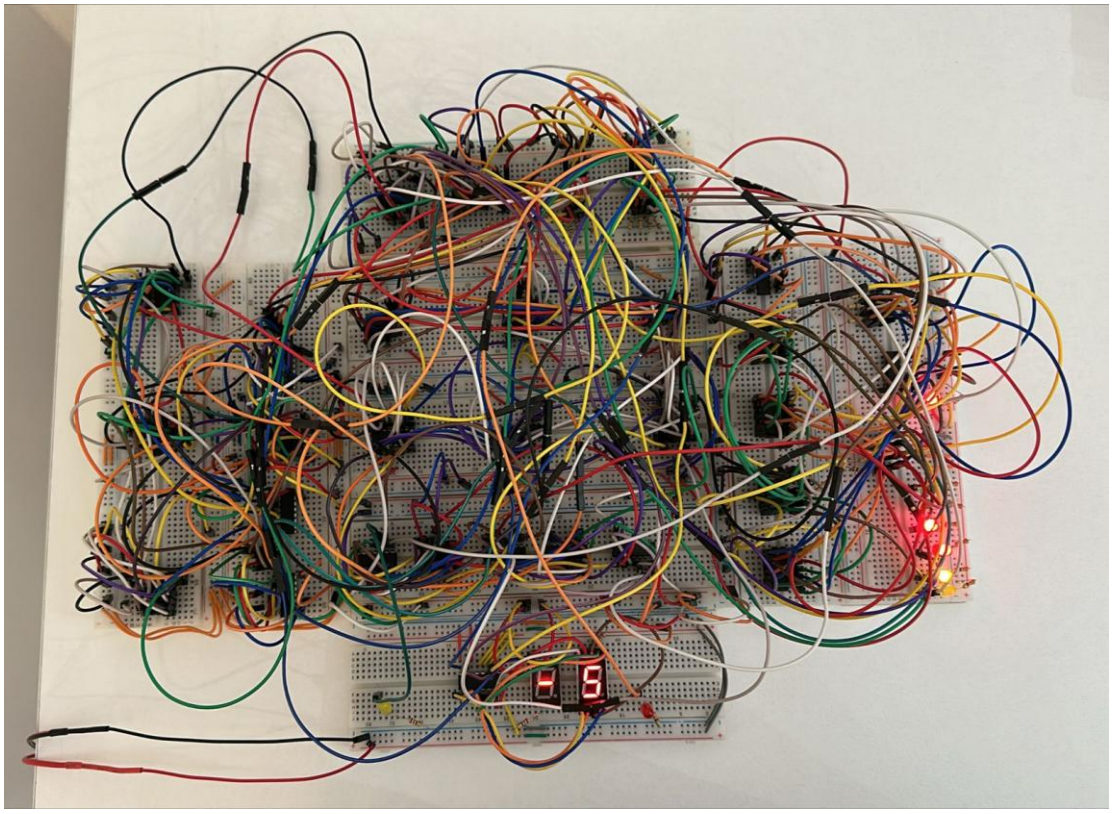


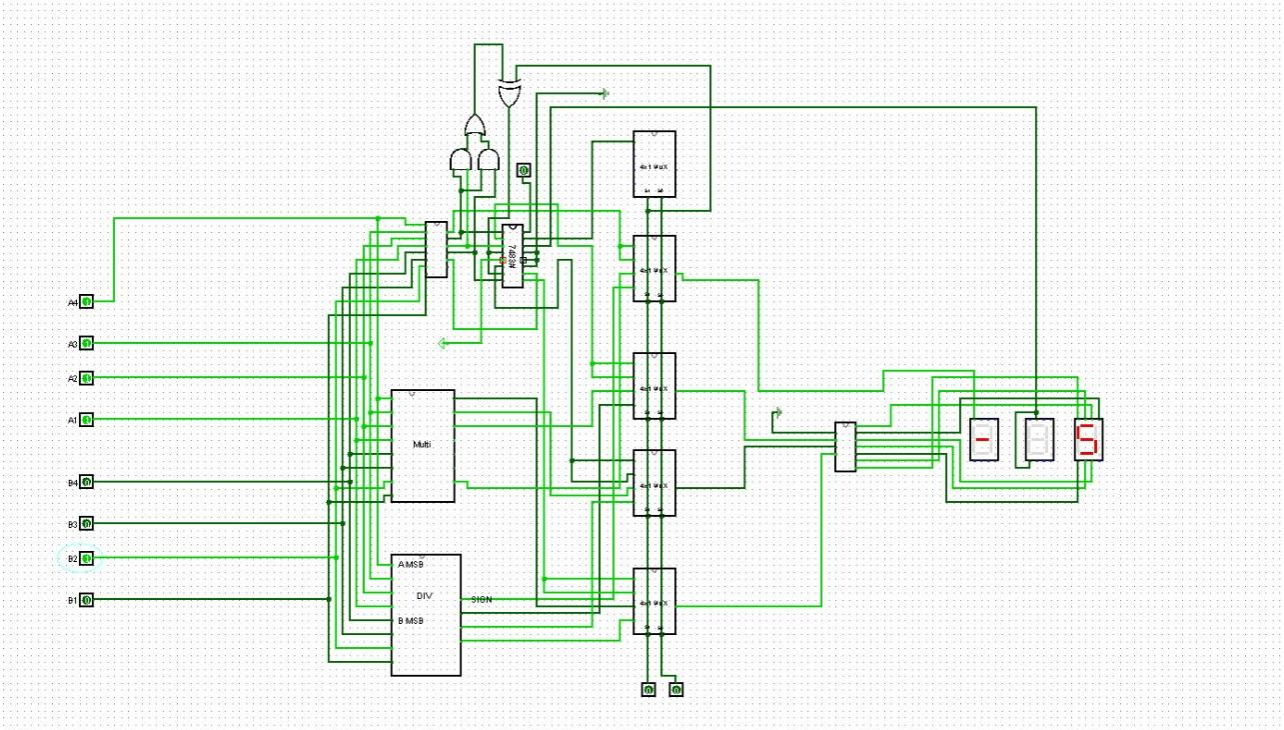
- Case (7+2) Overflow Red Led



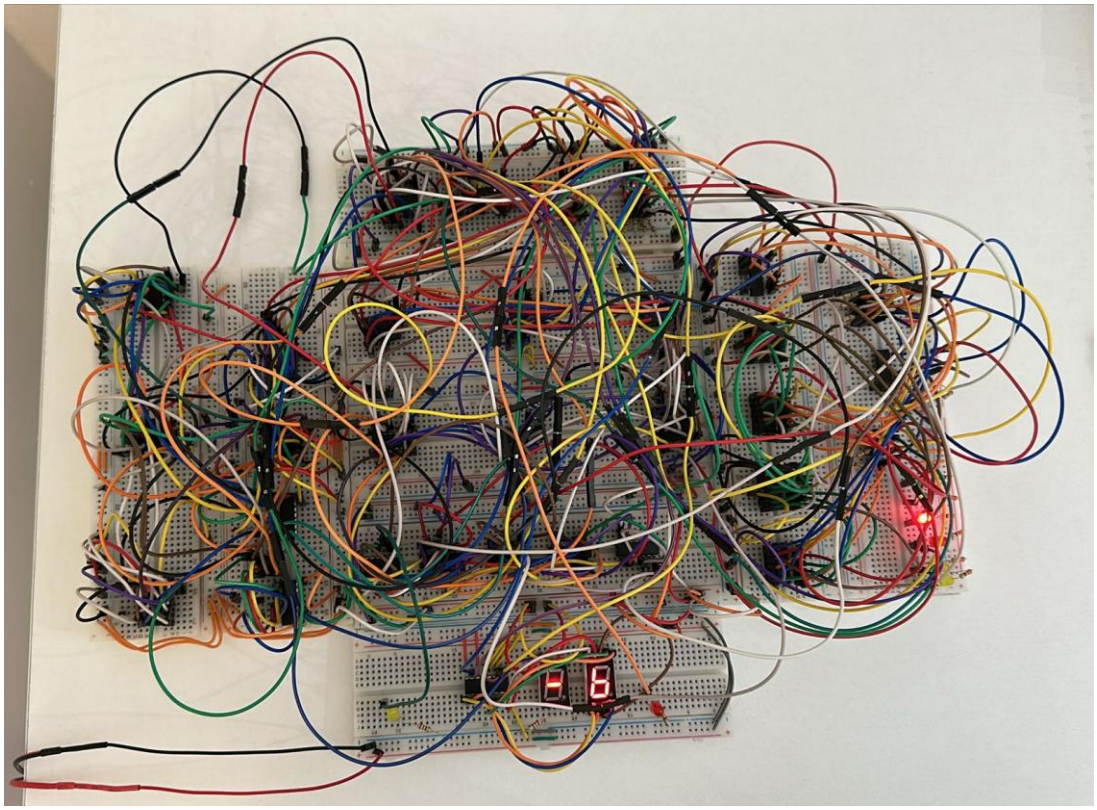


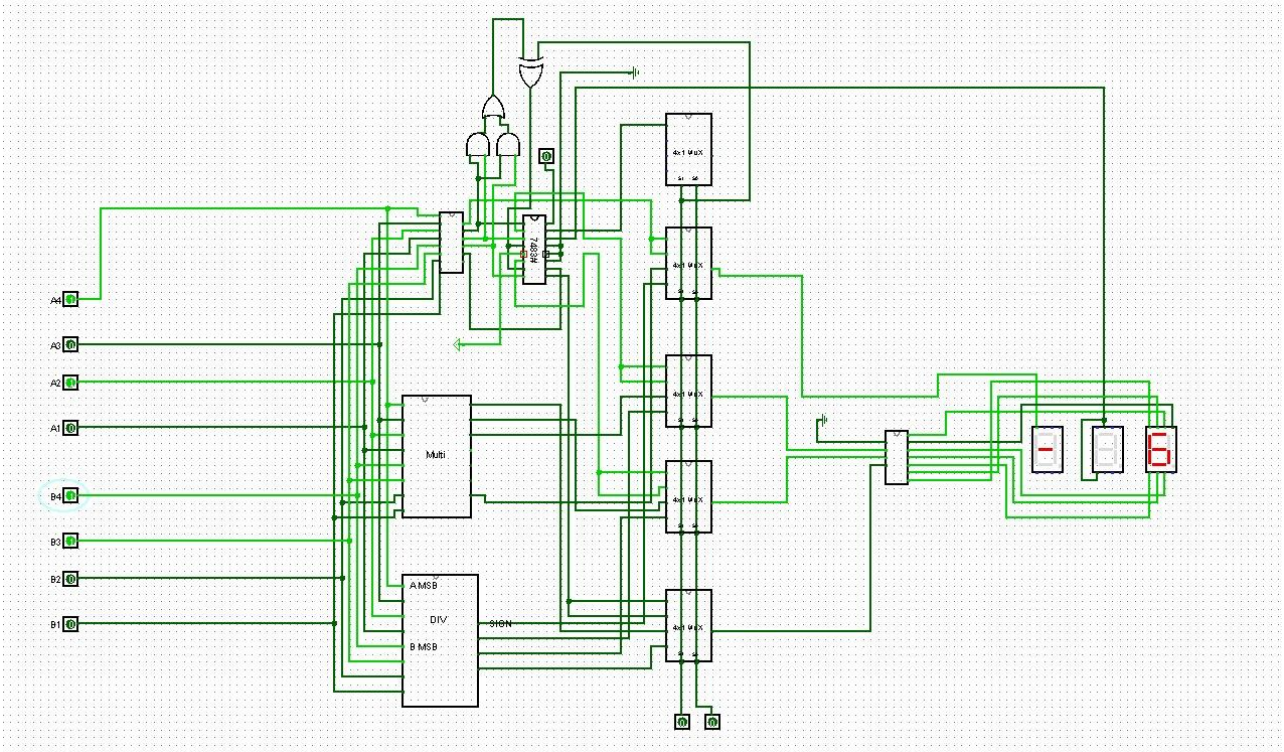
- Case (-7+2)



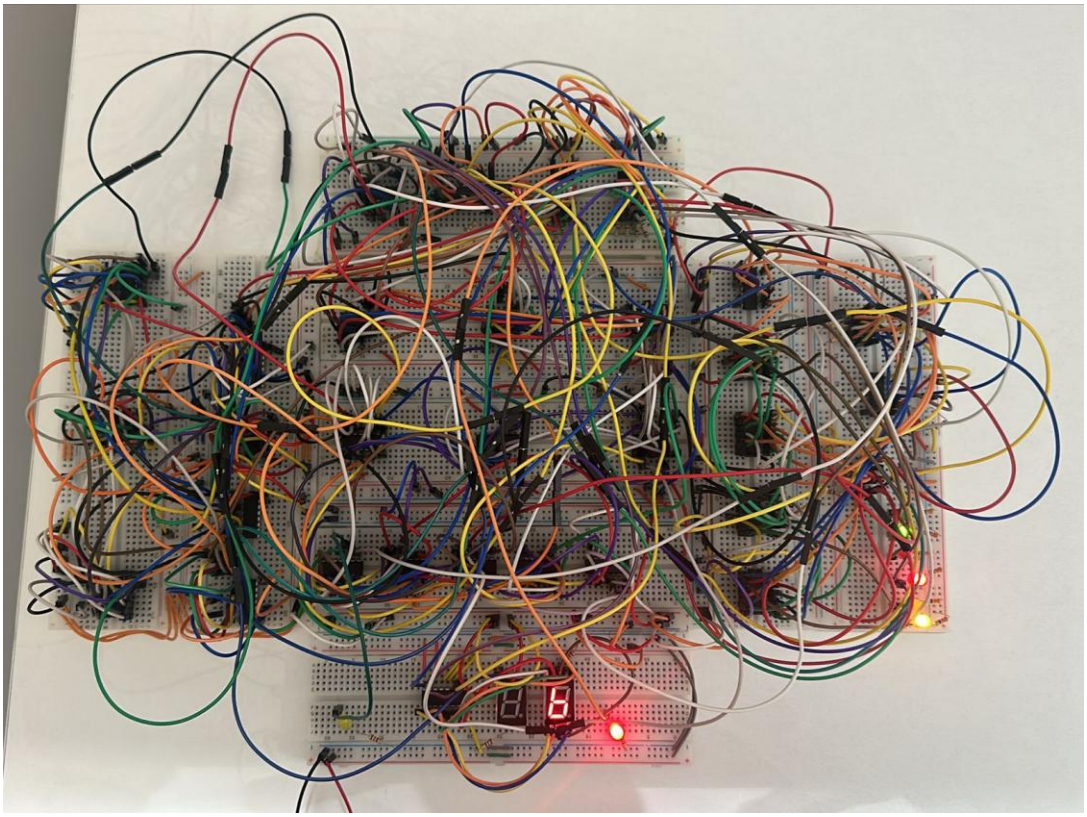


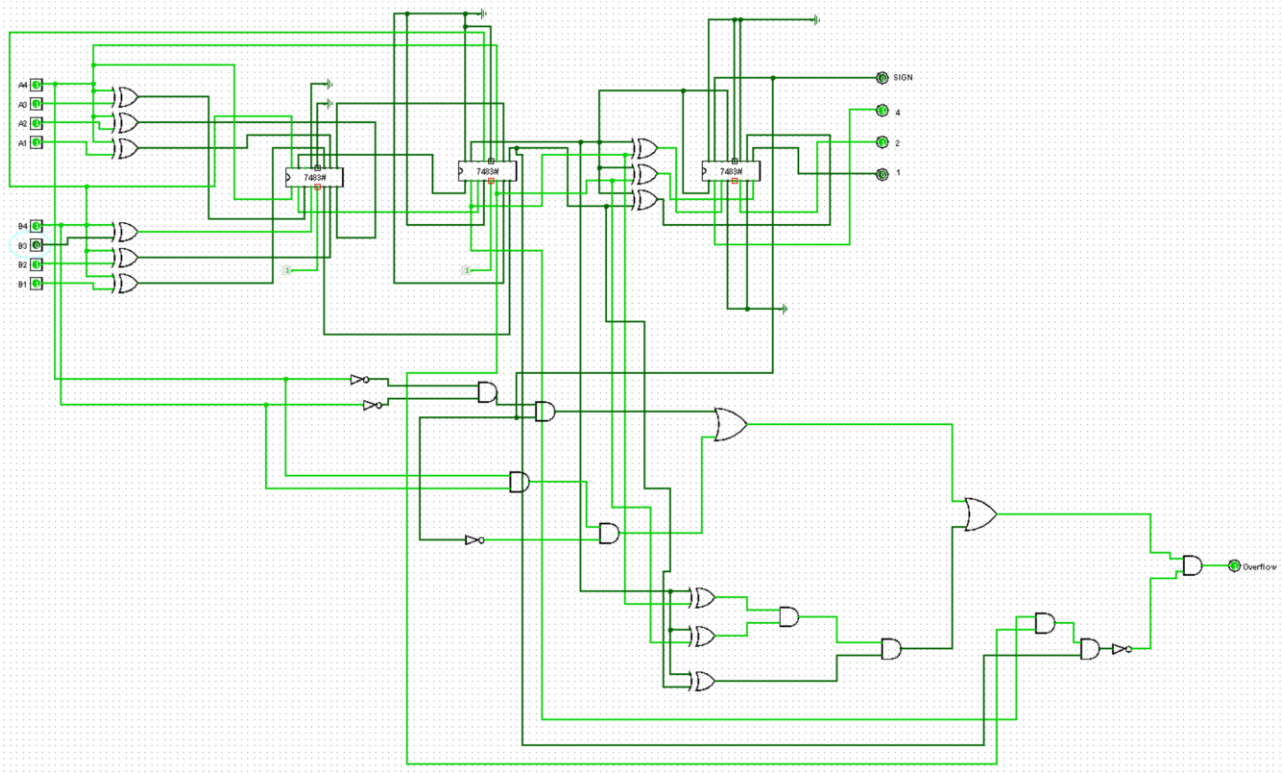
- Case (-2-4)




A4
A3
A2
A1
B4
B3
B2
B1
Multi
A MSB
DIV
B MSB
SIGN
4x1 MUX
4x1 MUX
4x1 MUX
4x1 MUX
4x1 MUX

- Case (-3-7) Overflow Red Led





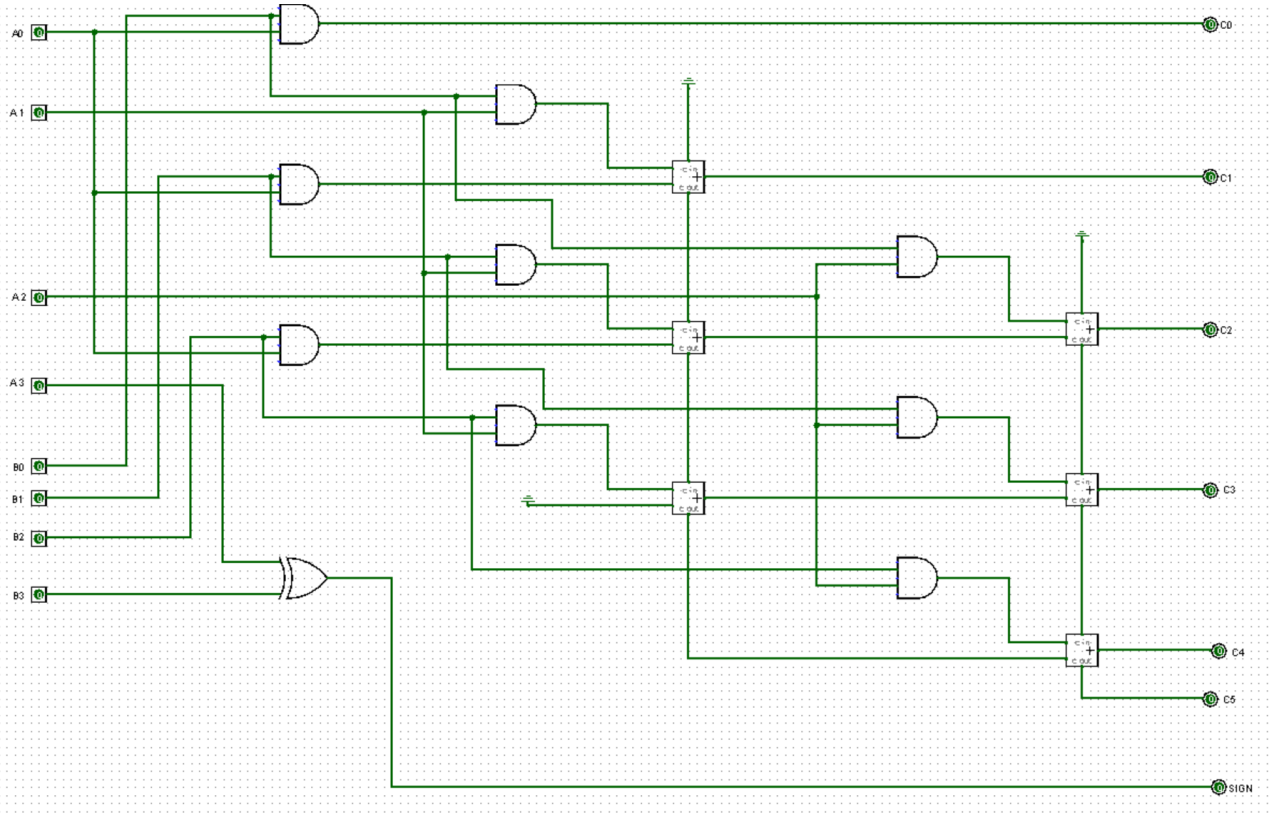
5.0 Multiplication:

For the multiplication operation, the circuit is activated when the multiplexer control input is set to 10. The ALU performs a 4-bit × 4-bit signed multiplication; however, since one bit in each operand is reserved for the sign, the effective operation becomes a 3-bit × 3-bit multiplication. This produces a 6-bit signed result. In the actual hardware schematic, we implemented two output display methods: first, we displayed the full 6-bit binary result on six LEDs along with an additional LED to indicate the sign; second, we configured the output to display results in the range of -7 to +7 using a seven-segment display, which allowed us to visualize small, signed multiplication results directly on hardware.

To extend the display capability and show the complete multiplication range, we designed a separate BCD conversion circuit in simulation. This circuit converts the 6-bit signed binary result into an 8-bit BCD format, taking into account the sign bit, which allowed us to accurately represent results from -49 to +49 on two seven-segment displays. This BCD circuit was built and tested separately in simulation and connected to the multiplication output to ensure the full functionality of the display.

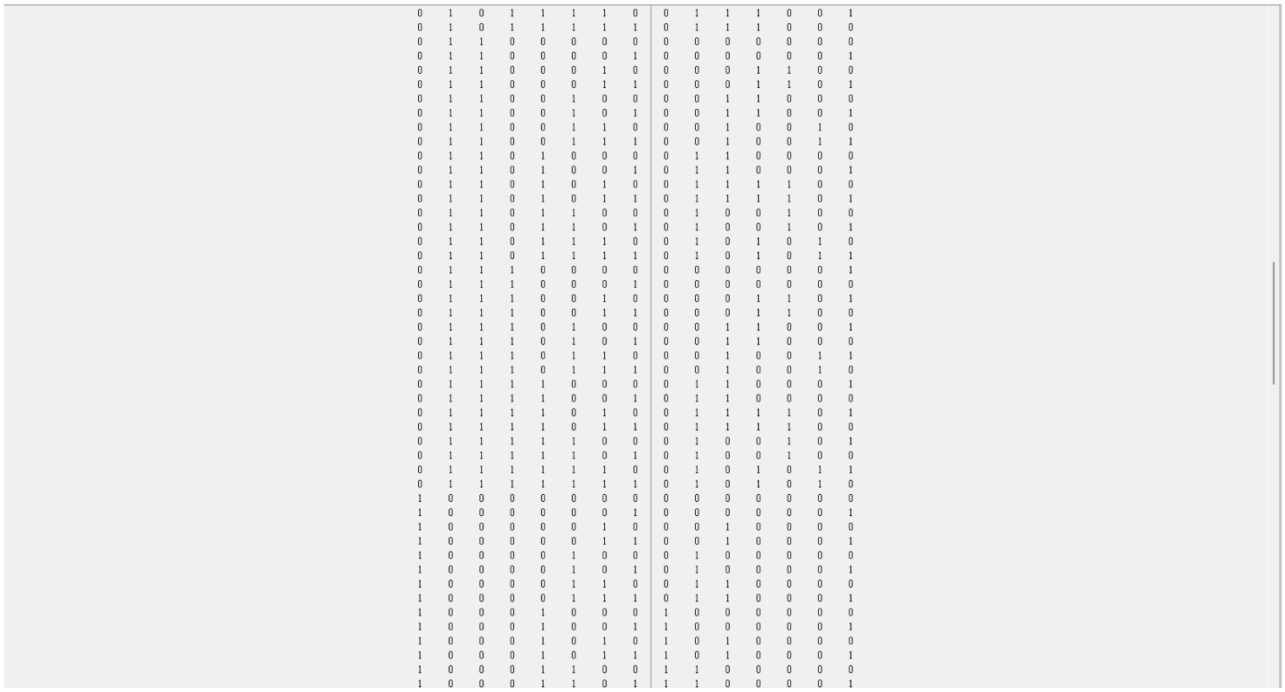
By combining both hardware implementation and simulation, we were able to verify the correctness of the multiplication operation, observe the result in multiple output forms, and explore different strategies for signed number representation in digital systems.

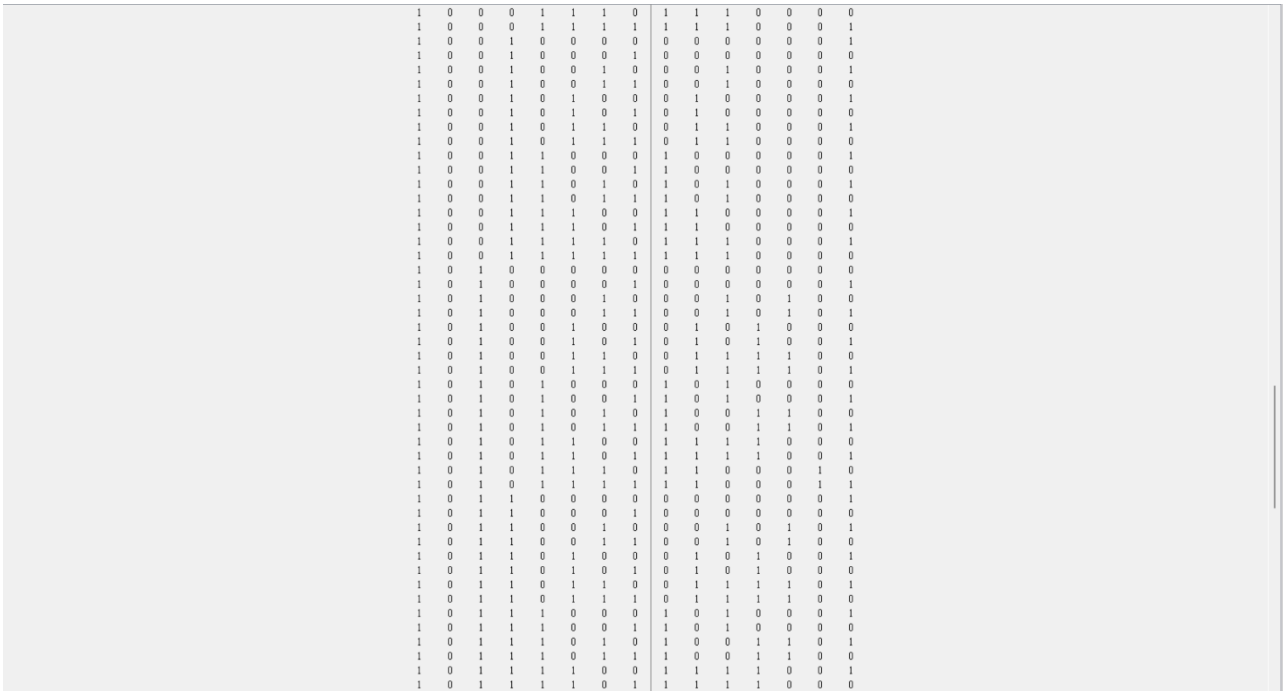
5.1. Multiplication Circuit:

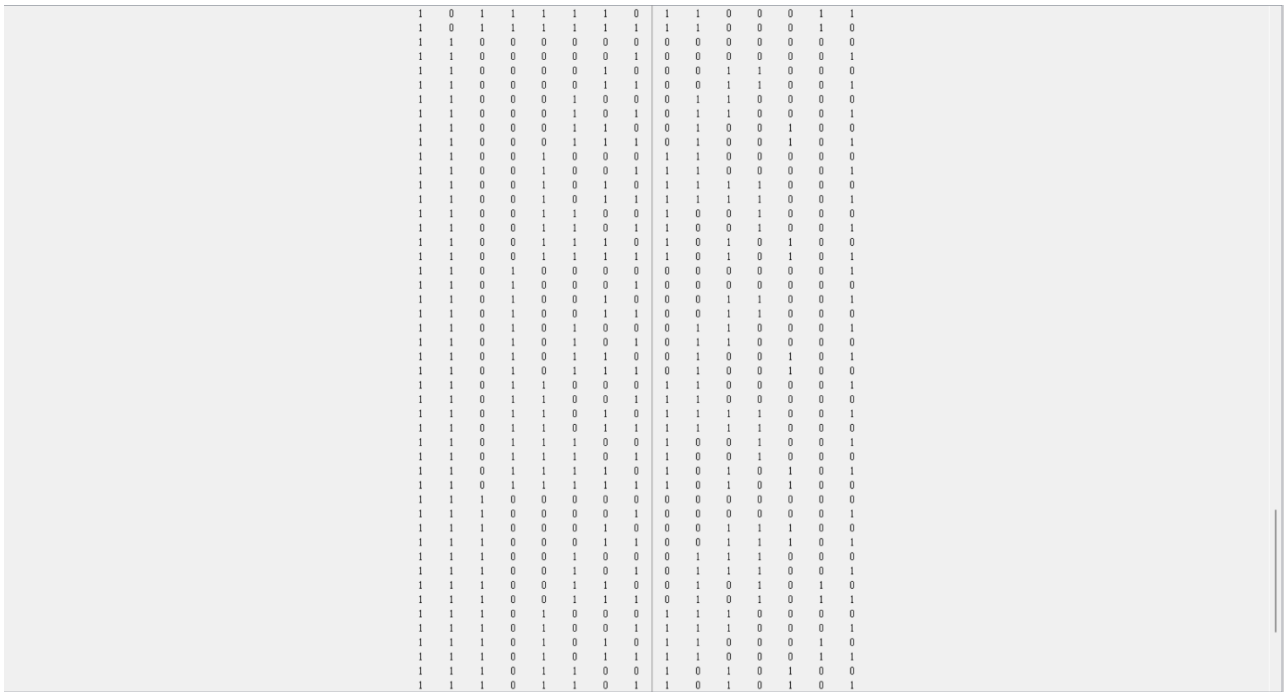


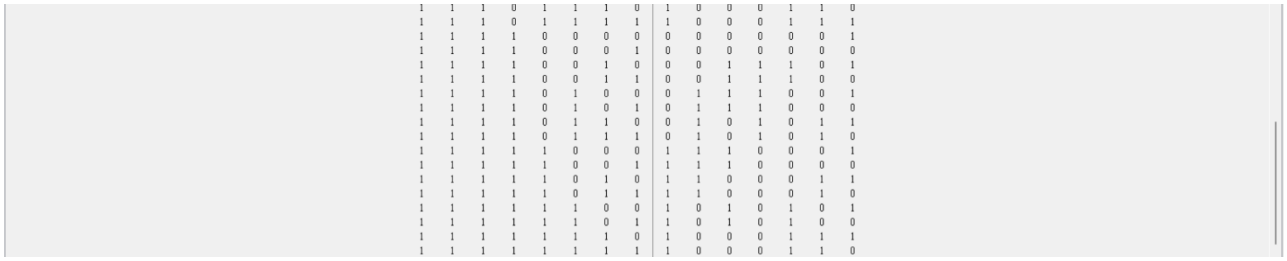
5.2. Truth Table:

A0	A1	A2	A3	B0	B1	B2	B3	C0	C1	C2	C3	C4	C5	SIGN
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1	0	0	0	0	0	0	1
0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	0	0	0	0	0	0	1
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	1	0	0	0	0	0	0	1
0	0	0	0	1	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	1	0	0	0	0	0	0	1
0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	1
0	0	0	1	0	1	0	1	0	0	0	0	0	0	0
0	0	0	1	0	1	1	0	0	0	0	0	0	0	1
0	0	0	1	0	1	1	1	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	1
0	0	0	1	1	0	0	1	0	0	0	0	0	0	0
0	0	0	1	1	0	1	0	0	0	0	0	0	0	1
0	0	0	1	1	0	1	1	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	1
0	0	0	1	1	1	0	1	0	0	0	0	0	0	0
0	0	0	1	1	1	1	0	0	0	0	0	0	0	1
0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
0	0	1	0	0	0	1	0	0	0	0	0	1	0	0
0	0	1	0	0	0	1	1	0	0	0	0	1	0	1
0	0	1	0	0	1	0	0	0	0	0	1	0	0	0
0	0	1	0	0	1	0	1	0	0	0	1	0	0	1
0	0	1	0	0	1	1	0	0	0	0	1	1	0	0
0	0	1	0	0	1	1	1	0	0	0	1	1	0	1
0	0	1	0	1	0	0	0	0	0	1	0	0	0	0
0	0	1	0	1	0	0	1	0	0	1	0	0	0	1
0	0	1	0	1	0	1	0	0	0	1	0	1	0	0
0	0	1	0	1	0	1	1	0	0	1	0	1	0	1
0	0	1	0	1	1	0	0	0	0	1	1	0	0	0
0	0	1	0	1	1	0	1	0	0	1	1	0	0	1
0	0	1	0	1	1	1	0	0	0	1	1	1	0	0
0	0	1	0	1	1	1	1	0	0	1	1	1	0	1
0	0	1	1	0	0	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0
0	0	1	1	0	0	1	0	0	0	0	0	1	0	1
0	0	1	1	0	0	1	1	0	0	0	0	1	0	0
0	0	1	1	0	1	0	0	0	0	0	1	0	0	1
0	0	1	1	0	1	0	1	0	0	0	1	0	0	0
0	0	1	1	0	1	1	0	0	0	0	1	1	0	1
0	0	1	1	0	1	1	1	0	0	0	1	1	0	0
0	0	1	1	1	0	0	0	0	0	1	0	0	0	1
0	0	1	1	1	0	0	1	0	0	1	0	0	0	0
0	0	1	1	1	0	1	0	0	0	1	0	1	0	1
0	0	1	1	1	0	1	1	0	0	1	0	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	0	0	1
0	0	1	1	1	1	0	1	0	0	1	1	0	0	0
0	0	1	1	1	1	1	0	0	0	1	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1	1	1	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0	0	1
0	1	0	0	0	0	1	0	0	0	0	1	0	0	0
0	1	0	0	0	0	1	1	0	0	0	1	0	0	1
0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
0	1	0	0	0	1	0	1	0	0	1	0	0	0	1
0	1	0	0	0	1	1	0	0	0	1	1	0	0	0
0	1	0	0	0	1	1	1	0	0	1	1	0	0	1
0	1	0	0	1	0	0	0	0	1	0	0	0	0	0
0	1	0	0	1	0	0	1	0	1	0	0	0	0	1
0	1	0	0	1	0	1	0	0	1	0	1	0	0	0
0	1	0	0	1	0	1	1	0	1	0	1	0	0	1
0	1	0	0	1	1	0	0	0	1	1	0	0	0	0
0	1	0	0	1	1	0	1	0	1	1	0	0	0	1
0	1	0	0	1	1	1	0	0	1	1	1	0	0	0
0	1	0	0	1	1	1	1	0	1	1	1	0	0	1
0	1	0	1	0	0	0	0	0	0	0	0	0	0	1
0	1	0	1	0	0	0	1	0	0	0	0	0	0	0
0	1	0	1	0	0	1	0	0	0	0	1	0	0	1
0	1	0	1	0	0	1	1	0	0	0	1	0	0	0
0	1	0	1	0	1	0	0	0	0	1	0	0	0	1
0	1	0	1	0	1	0	1	0	0	1	0	0	0	0
0	1	0	1	0	1	1	0	0	0	1	1	0	0	1
0	1	0	1	0	1	1	1	0	0	1	1	0	0	0
0	1	0	1	1	0	0	0	0	1	0	0	0	0	1
0	1	0	1	1	0	0	1	0	1	0	0	0	0	0
0	1	0	1	1	0	1	0	0	1	0	1	0	0	1
0	1	0	1	1	0	1	1	0	1	0	1	0	0	0
0	1	0	1	1	1	0	0	0	1	1	0	0	0	1
0	1	0	1	1	1	0	1	0	1	1	0	0	0	0

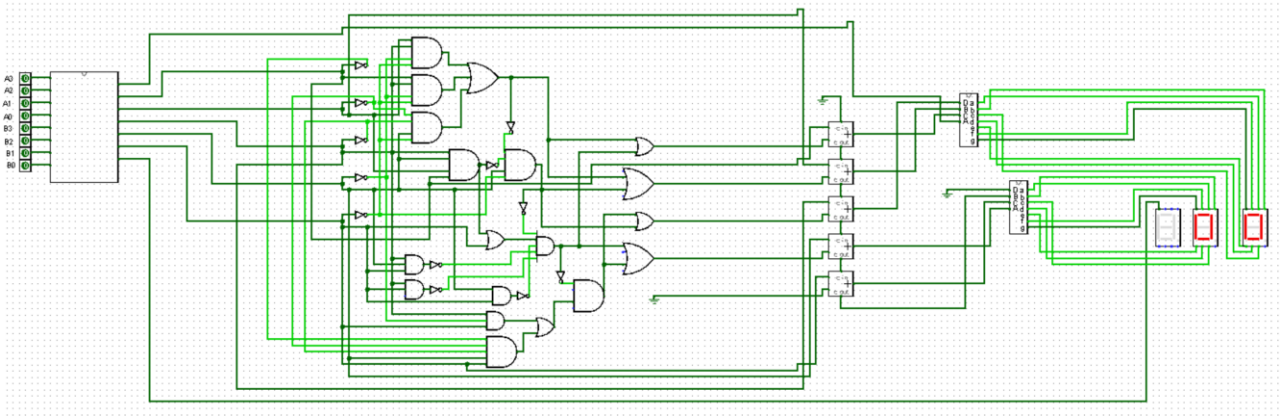






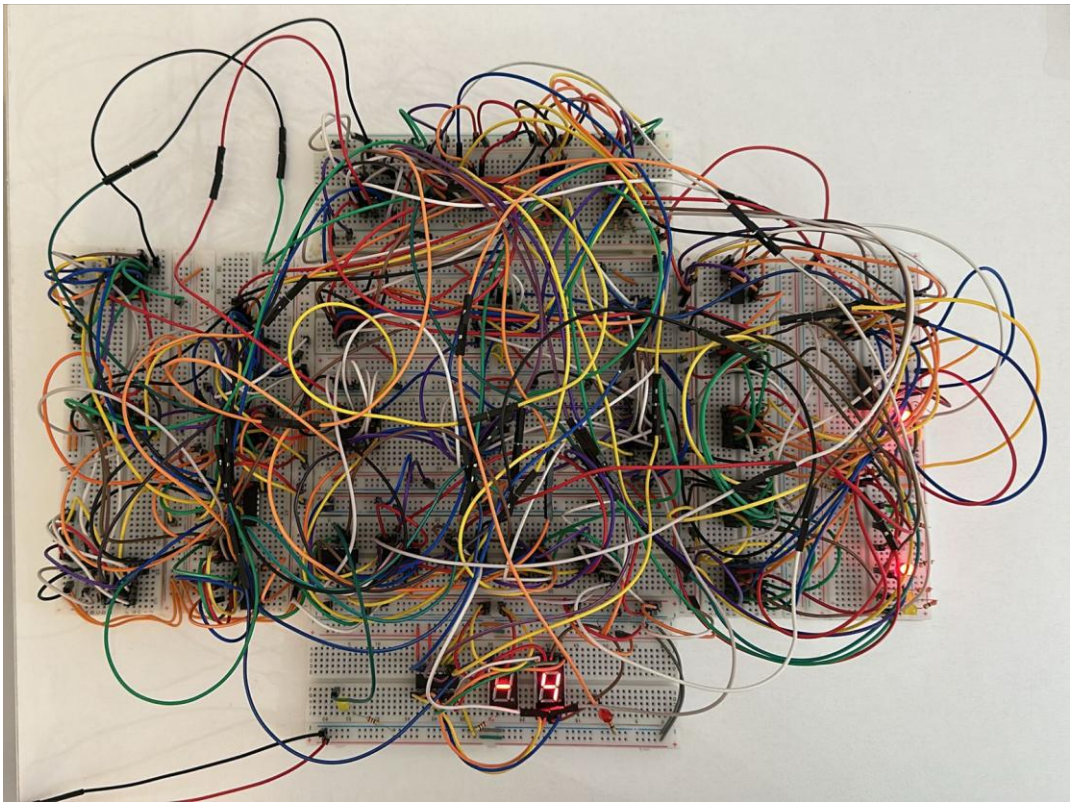


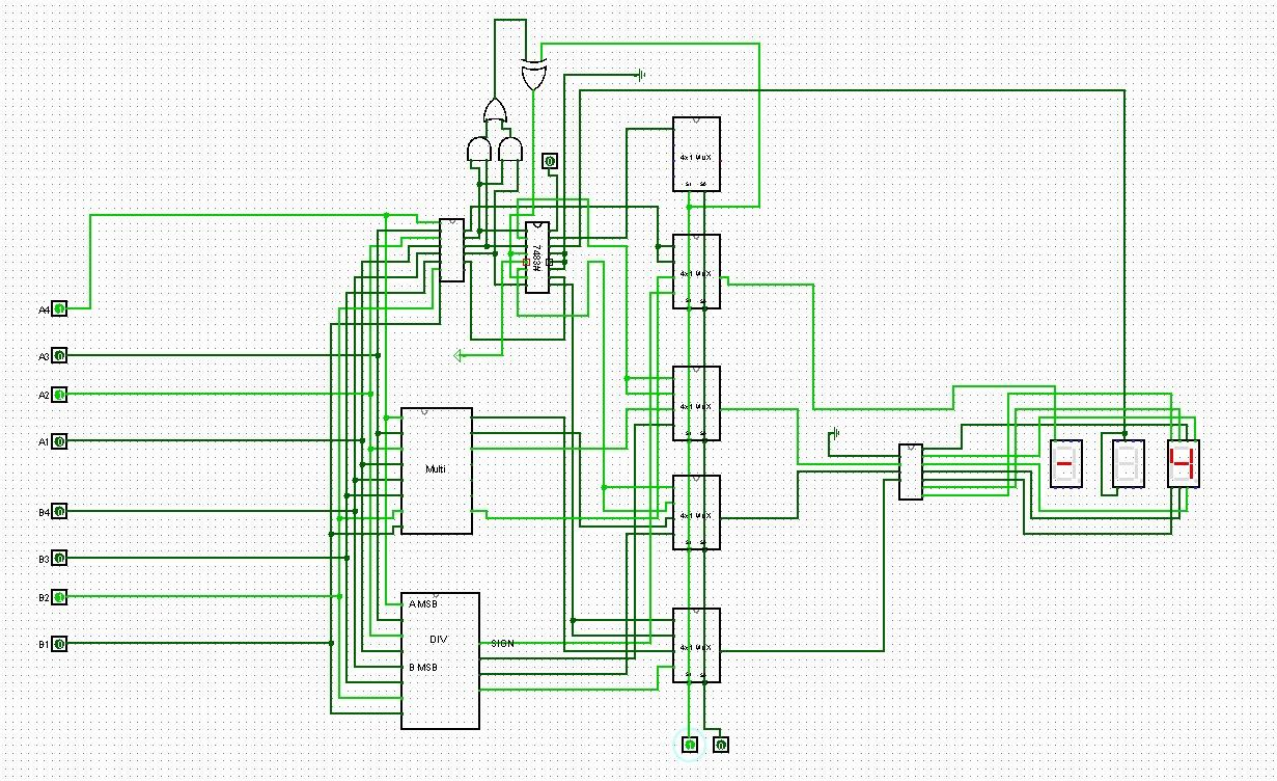
5.3. Multiplication BCD Convertor:



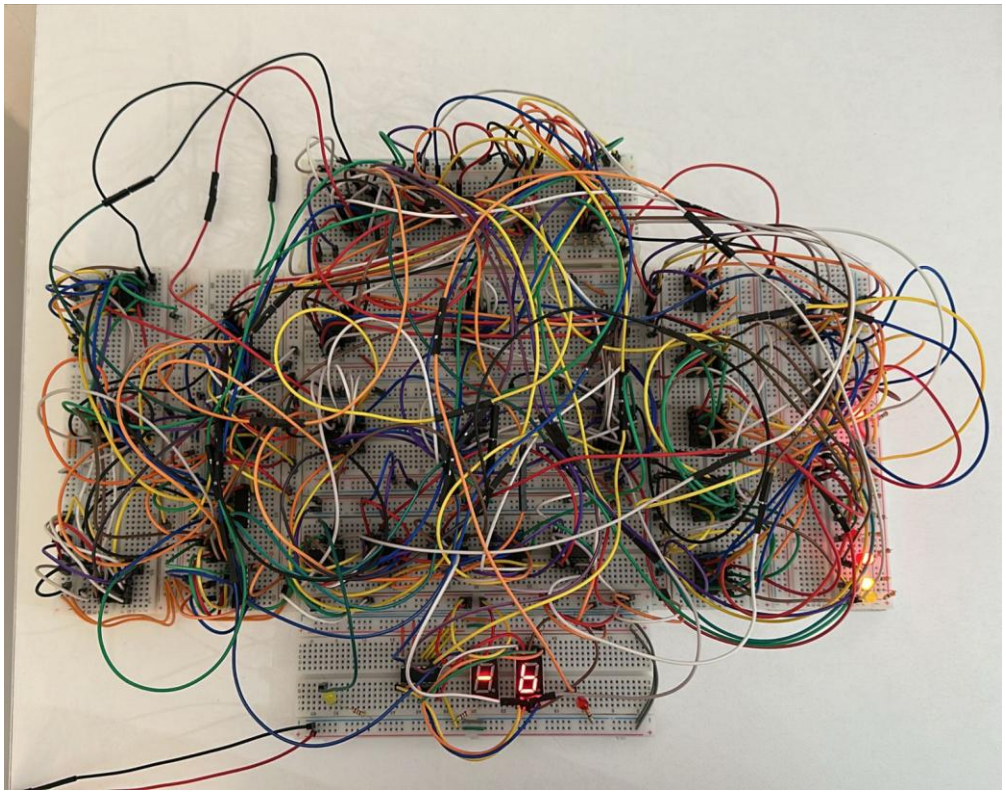
5.4. Cases

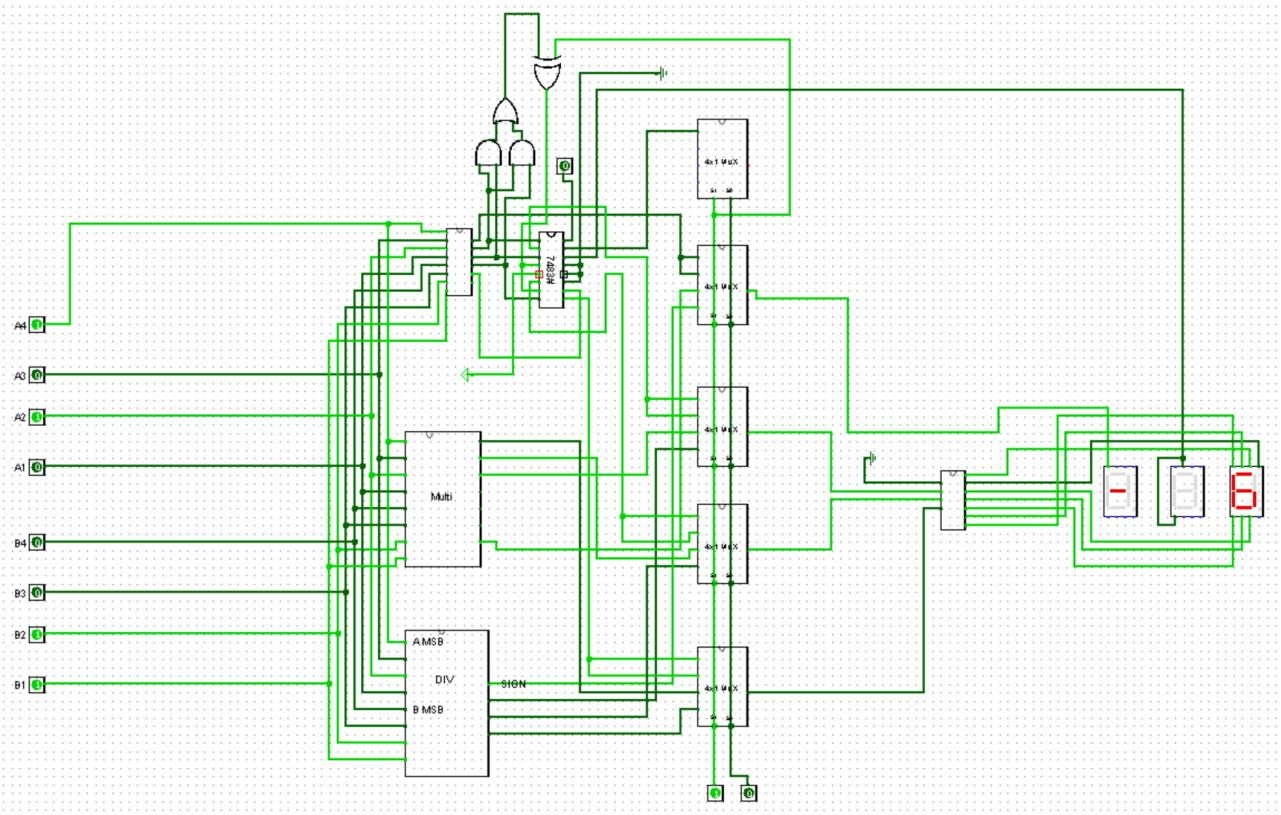
- Case (2 * -2)



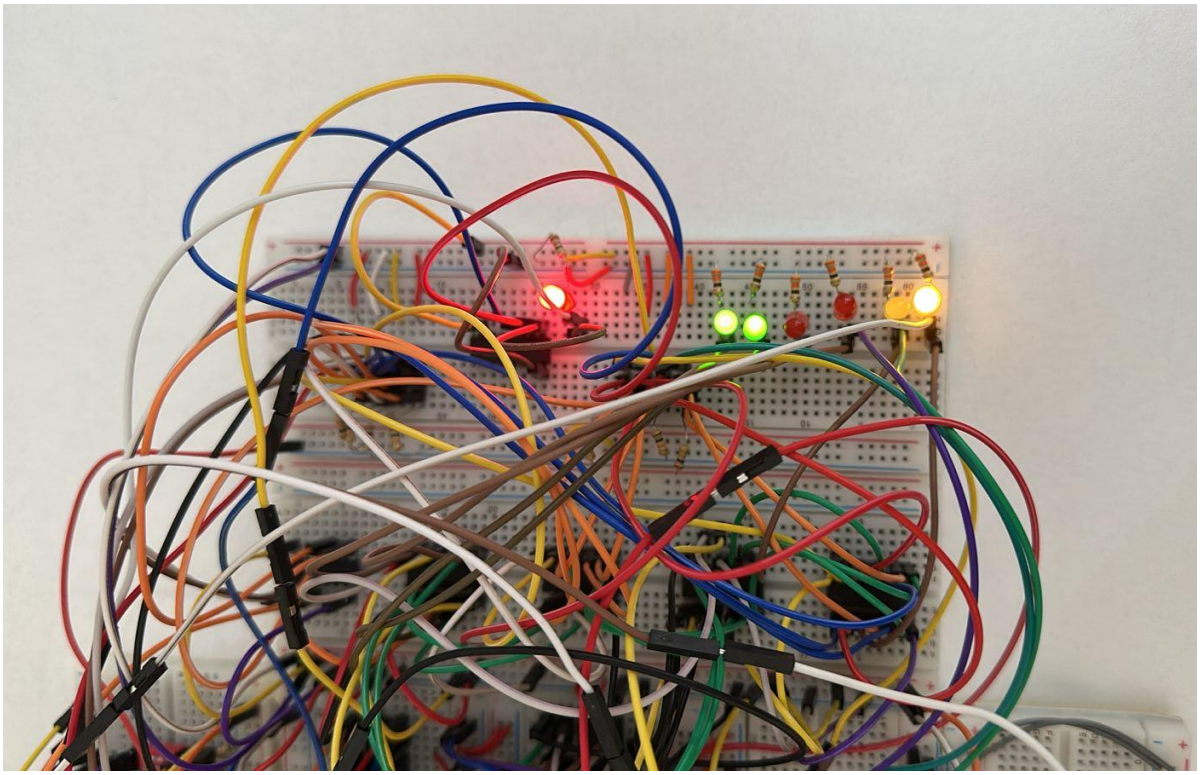


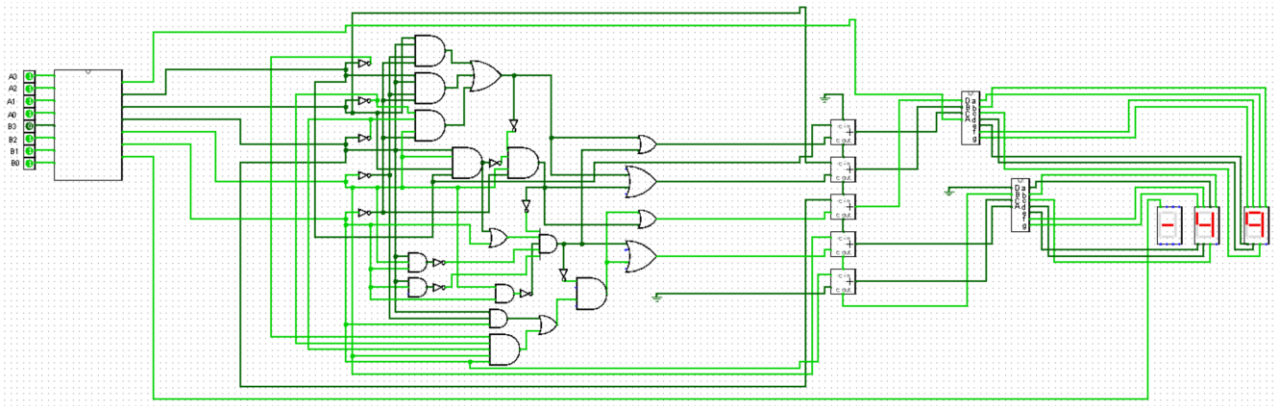
- Case (-2 * 3)



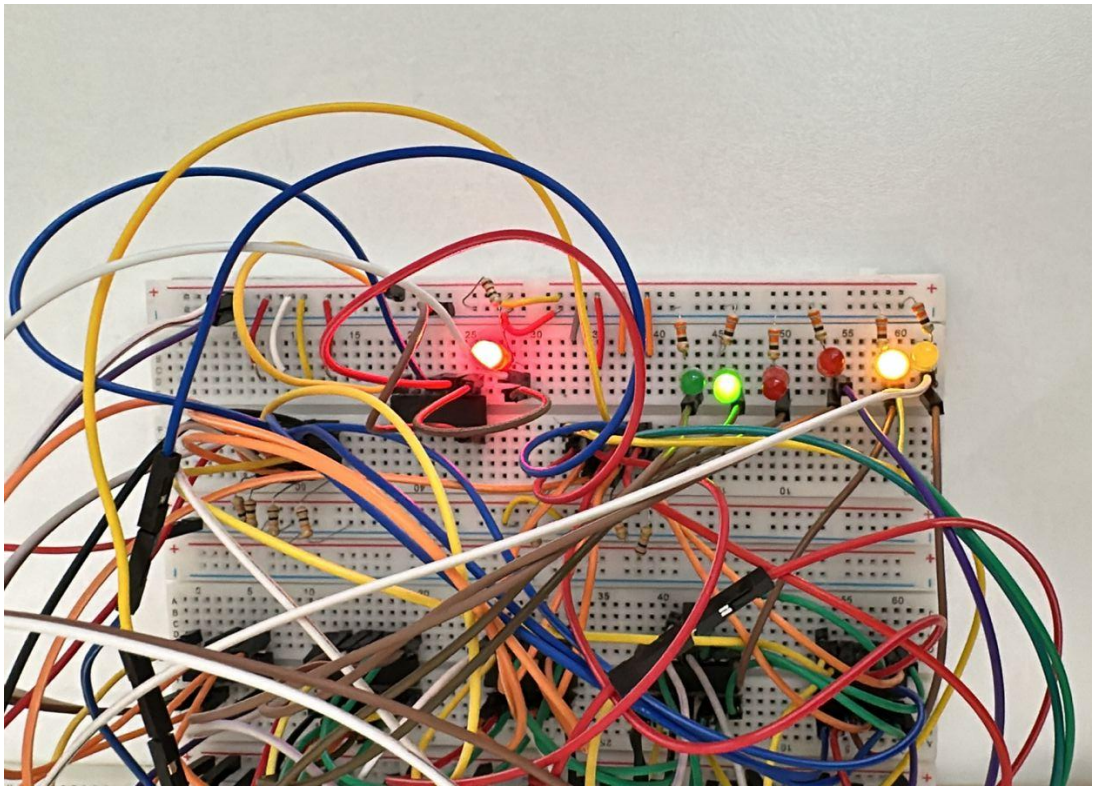

A4
A3
A2
A1
B4
B3
B2
B1
Multi
AMSB
DIV
BMSB
SIGN
4x1 MUX
4x1 MUX
4x1 MUX
4x1 MUX
4x1 MUX

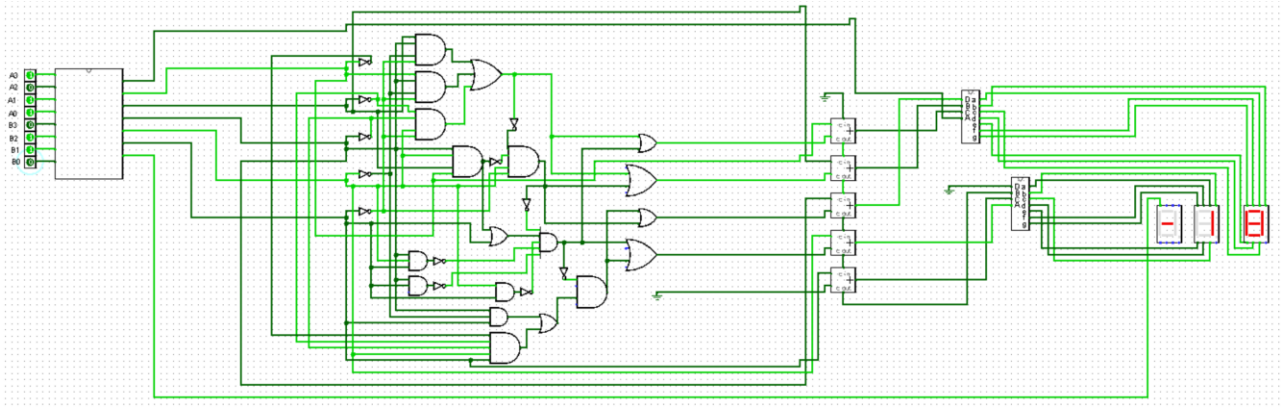
- Case (-7 * 7)





- Case (-3 * 6)





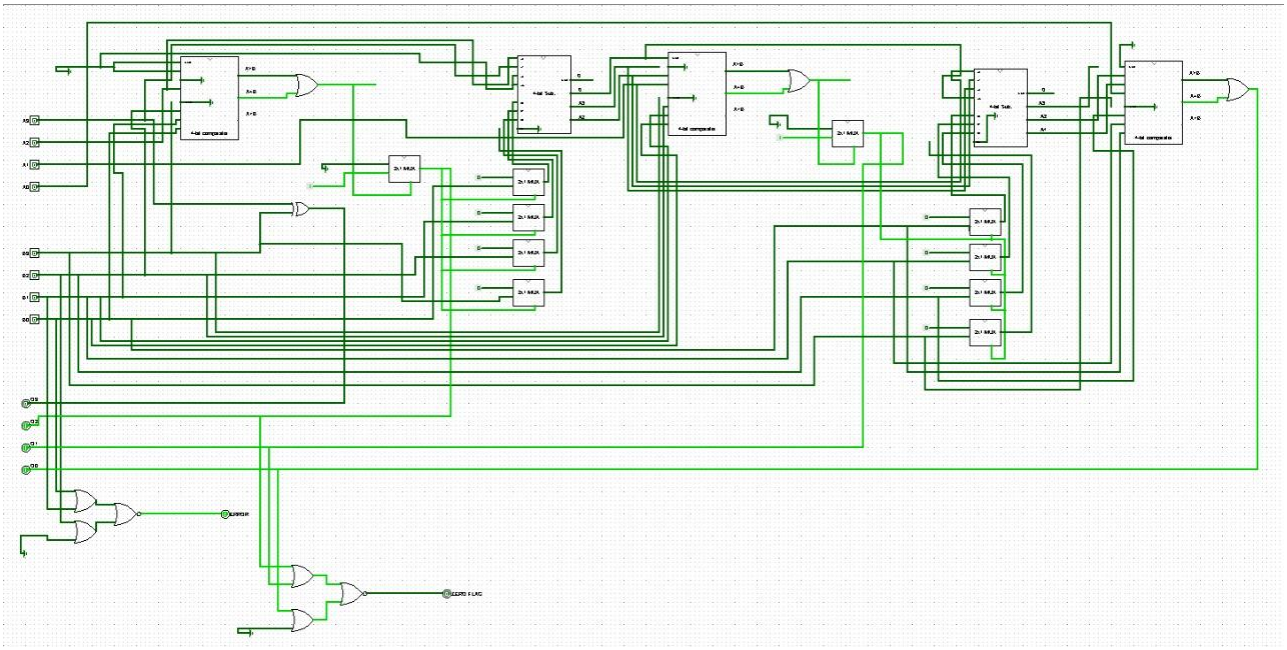
6.0 Division:

For the division operation, we designed a dedicated binary division circuit that performs signed division between two 4-bit numbers, A (dividend) and B (divisor), using the long division method. The circuit supports inputs in sign-magnitude representation and outputs a signed 4-bit quotient, with values ranging from -7 to +7. The operation is activated when the 2-bit control signal for the multiplexer is set to 11. This selection ensures that the division unit is enabled only when explicitly required, preventing conflicts with other arithmetic operations within the ALU.

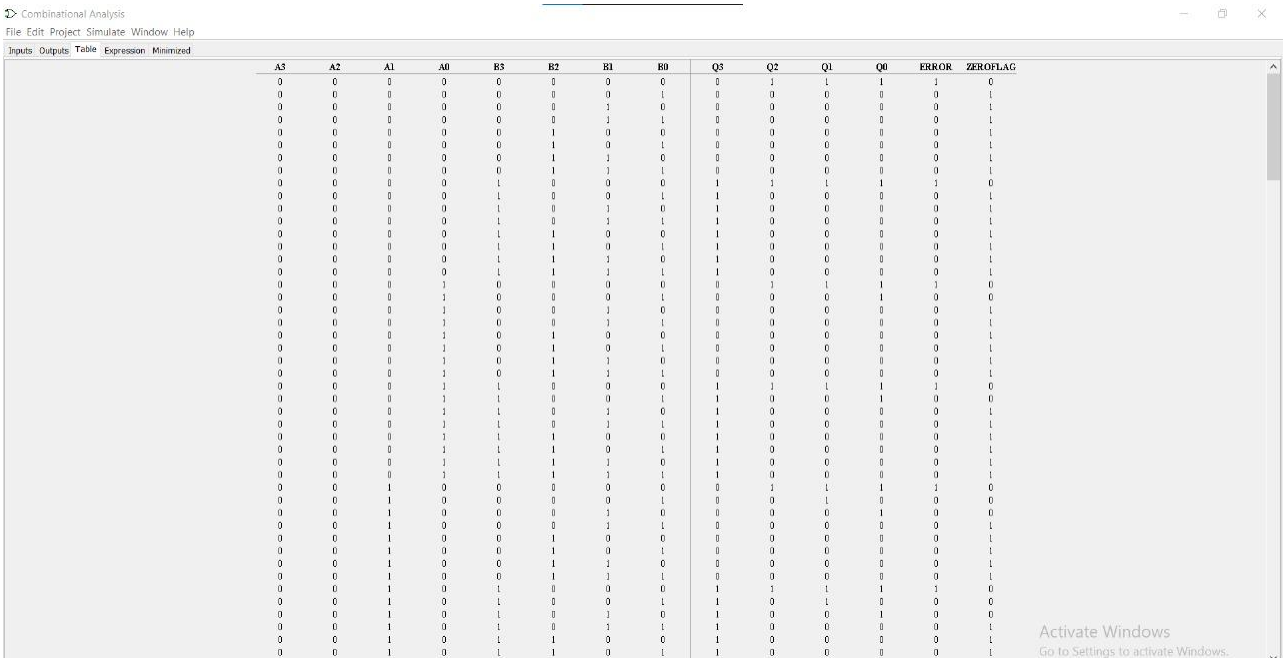
To implement the long division logic, we constructed the circuit using full adders and multiplexers. The control logic manages the shifting, subtraction, and sign adjustment required by the long division algorithm. The design carefully handles signed numbers by determining the signs of both operands and applying the appropriate sign to the final quotient. An important feature of the design is error detection for division by zero. If divisor B is zero, the circuit activates an error LED indicator to alert the user. This ensures safe operation and prevents undefined behavior in the hardware.

The hardware version of the circuit is limited to displaying quotients in the range -7 to +7, matching the constraints of our 4-bit signed output representation. Any inputs producing results outside this range are inherently bounded due to the bit-width, and an error condition is only flagged for division by zero. This standalone division circuit was tested and demonstrated on hardware using the specified control and indicator signals. The use of full adders and multiplexers allowed for a fully combinational implementation, consistent with digital logic design practices.

6.1. Division Circuit:



6.2. Truth Table:

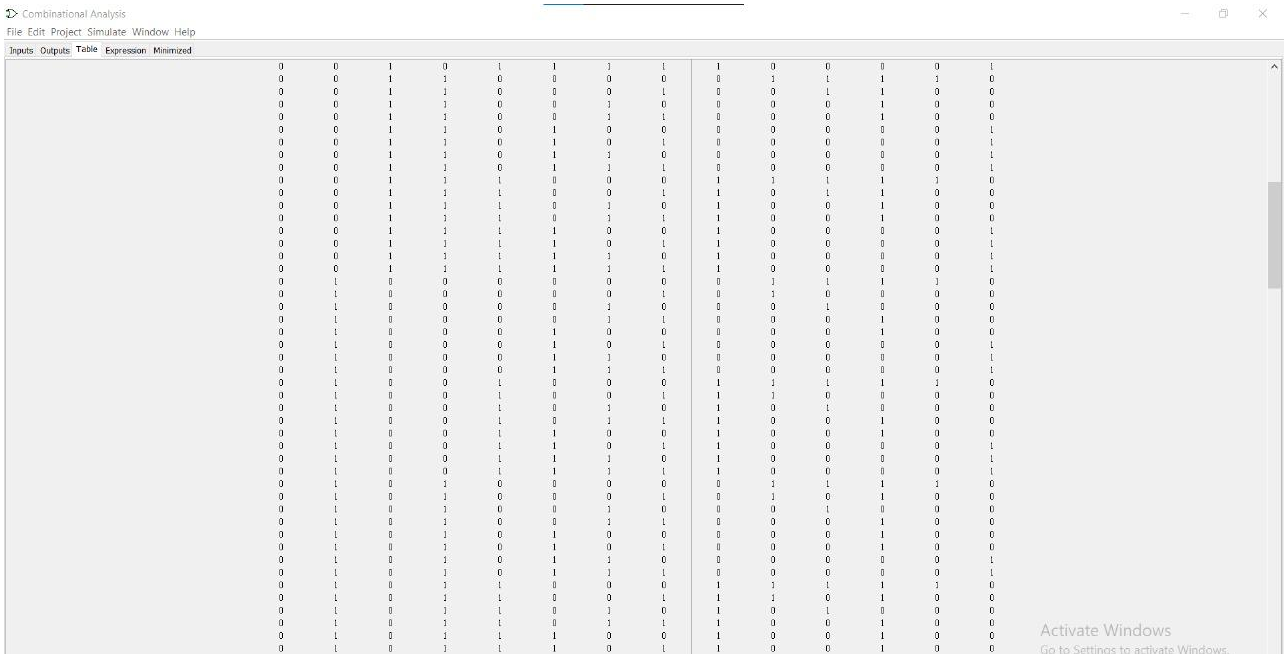


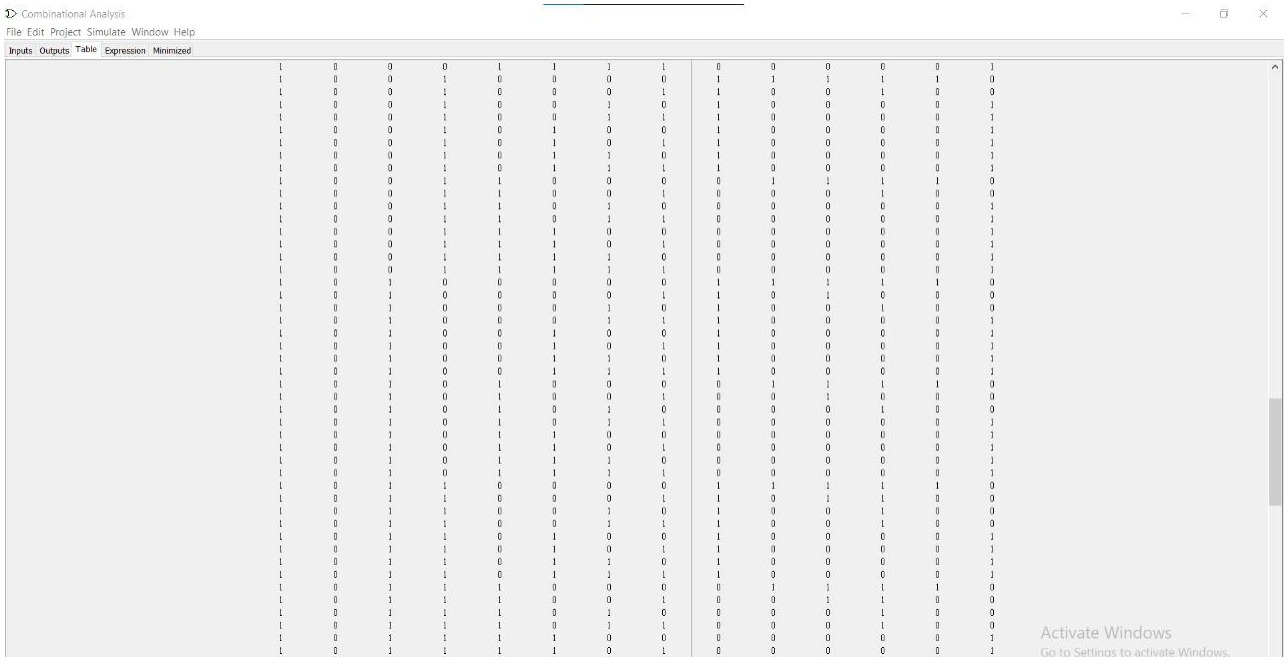
Combinational Analysis

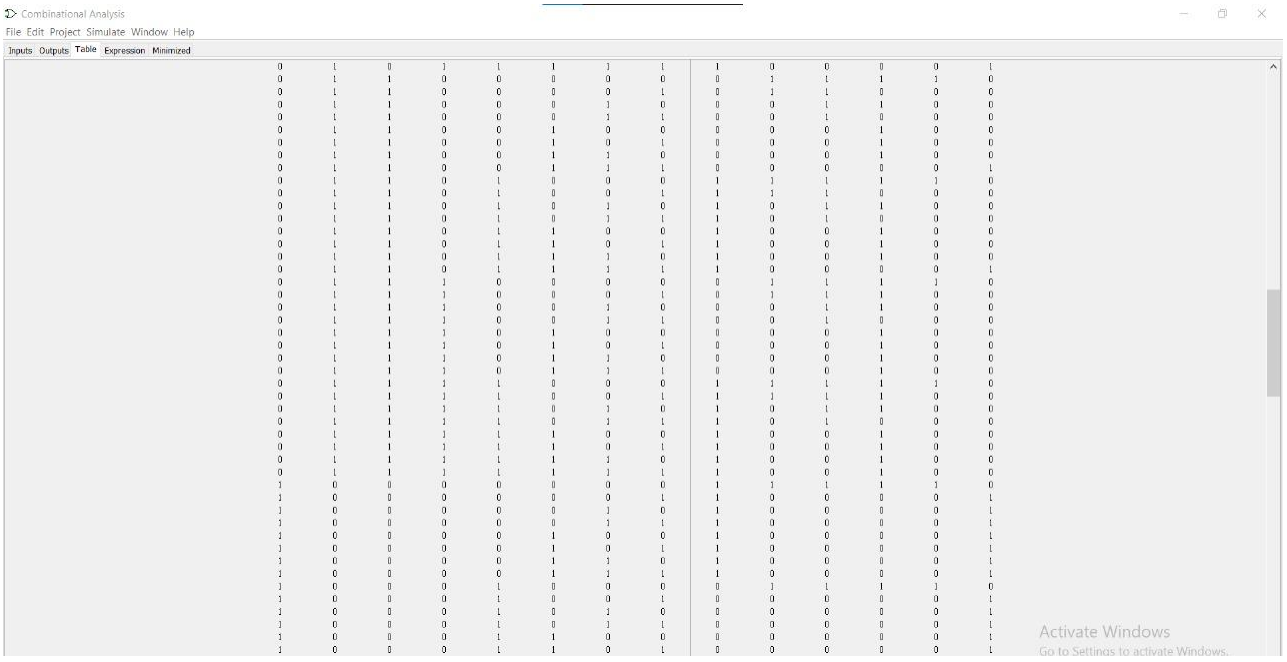
File Edit Project Simulate Window Help

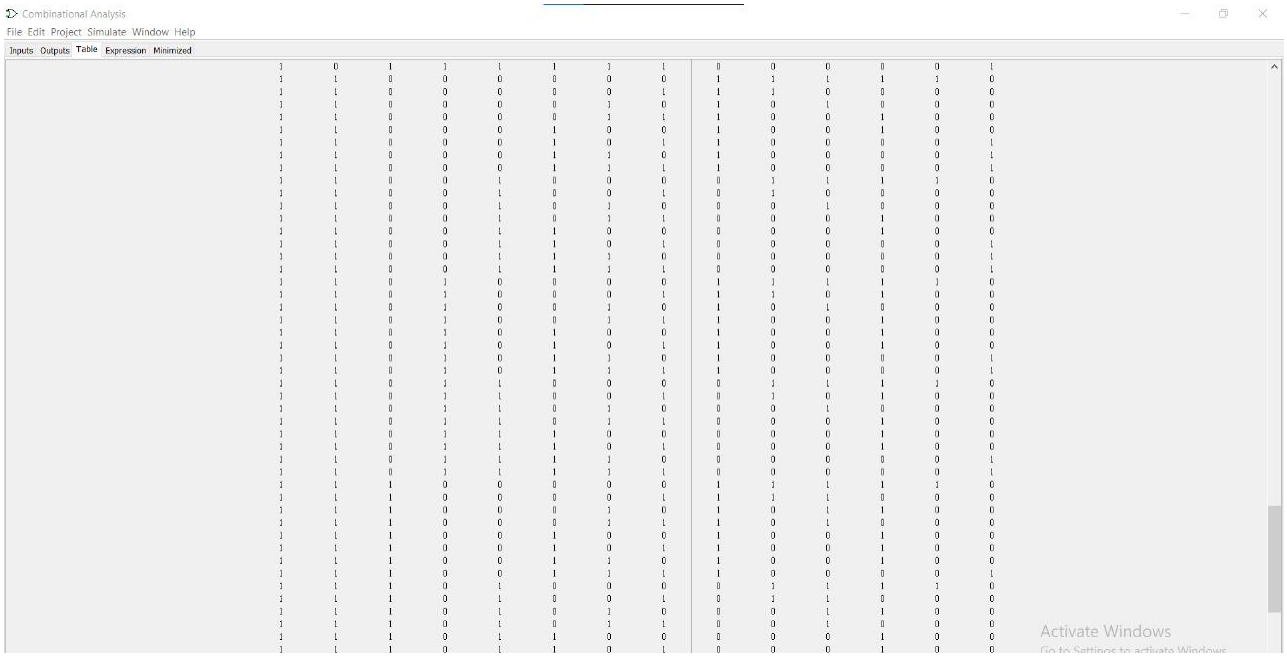
Inputs Outputs Table Expression Minimized

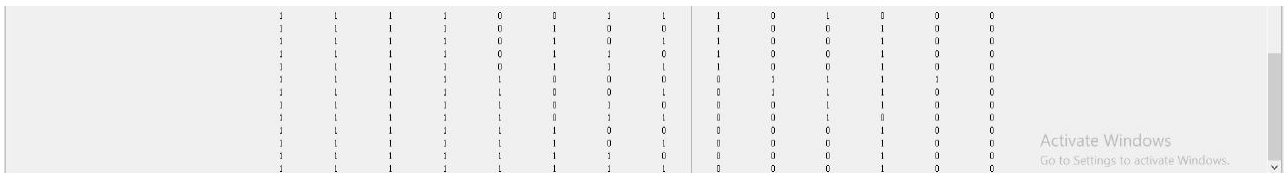
A3	A2	A1	A0	B3	B2	B1	B0	Q3	Q2	Q1	Q0	ERROR	ZEROFLAG
0	0	0	0	0	0	0	0	0	1	1	1	1	0
0	0	0	0	0	0	0	1	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1	0	0	0	0	0	1
0	0	0	0	0	1	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	1	0	0	0	0	0	1
0	0	0	0	0	1	1	0	0	0	0	0	0	1
0	0	0	0	0	1	1	1	0	0	0	0	0	1
0	0	0	0	1	0	0	0	1	1	1	1	1	0
0	0	0	0	1	0	0	1	1	0	0	0	0	1
0	0	0	0	1	0	1	0	1	0	0	0	0	1
0	0	0	0	1	0	1	1	1	0	0	0	0	1
0	0	0	0	1	1	0	0	1	0	0	0	0	1
0	0	0	0	1	1	0	1	1	0	0	0	0	1
0	0	0	0	1	1	1	0	1	0	0	0	0	1
0	0	0	0	1	1	1	1	1	0	0	0	0	1
0	0	0	1	0	0	0	0	0	1	1	1	1	0
0	0	0	1	0	0	0	1	0	0	0	1	0	0
0	0	0	1	0	0	1	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1	0	0	0	0	0	1
0	0	0	1	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	1	0	1	0	0	0	0	0	1
0	0	0	1	0	1	1	0	0	0	0	0	0	1
0	0	0	1	0	1	1	1	0	0	0	0	0	1
0	0	0	1	1	0	0	0	1	1	1	1	1	0
0	0	0	1	1	0	0	1	1	0	0	1	0	0
0	0	0	1	1	0	1	0	1	0	0	0	0	1
0	0	0	1	1	0	1	1	1	0	0	0	0	1
0	0	0	1	1	1	0	0	1	0	0	0	0	1
0	0	0	1	1	1	0	1	1	0	0	0	0	1
0	0	0	1	1	1	1	0	1	0	0	0	0	1
0	0	0	1	1	1	1	1	1	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	1	1	1	0
0	0	1	0	0	0	0	1	0	0	1	0	0	0
0	0	1	0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	1	1	0	0	0	0	0	1
0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	1	0	1	0	0	0	0	0	1
0	0	1	0	0	1	1	0	0	0	0	0	0	1
0	0	1	0	0	1	1	1	0	0	0	0	0	1
0	0	1	0	1	0	0	0	1	1	1	1	1	0
0	0	1	0	1	0	0	1	1	0	0	0	0	0
0	0	1	0	1	0	1	0	1	0	0	0	0	1
0	0	1	0	1	0	1	1	1	0	0	0	0	1
0	0	1	0	1	1	0	0	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	0	0	0	1


Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.


Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.

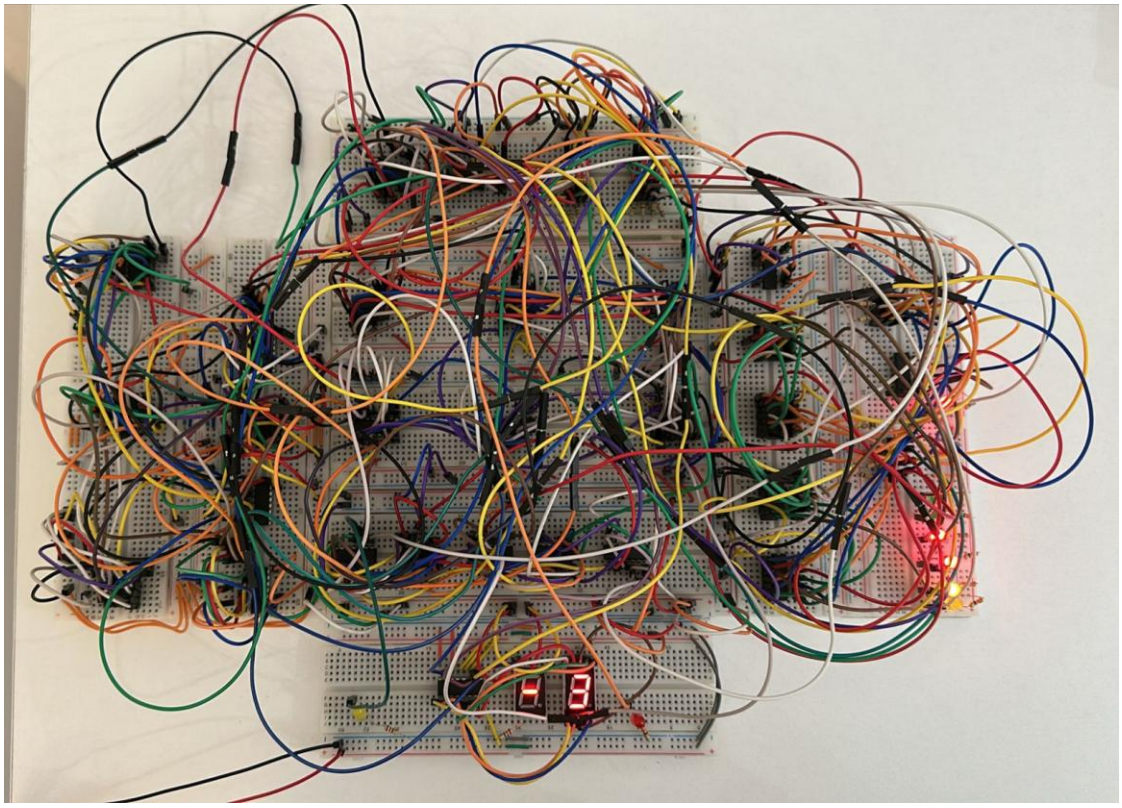

Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.

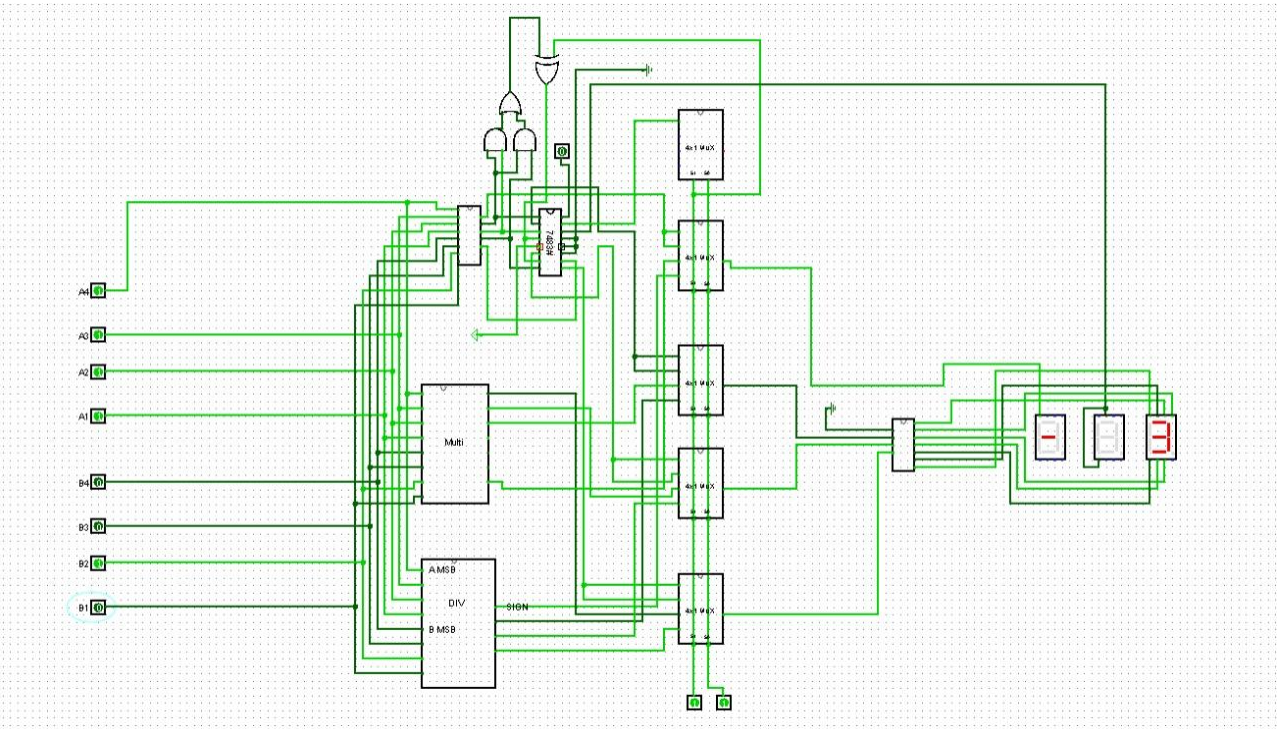

Combinational Analysis
File Edit Project Simulate Window Help
Inputs Outputs Table Expression Minimized
Activate Windows
Go to Settings to activate Windows.


Activate Windows
Go to Settings to activate Windows.

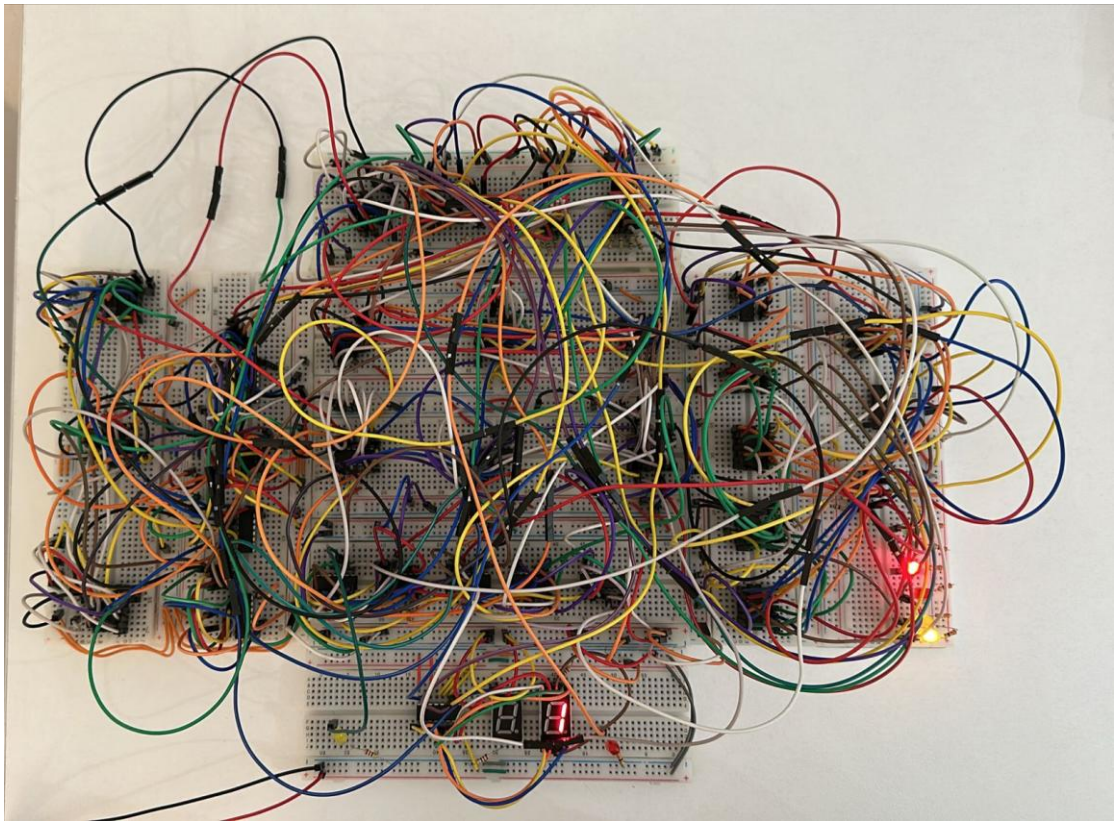
6.3. Cases

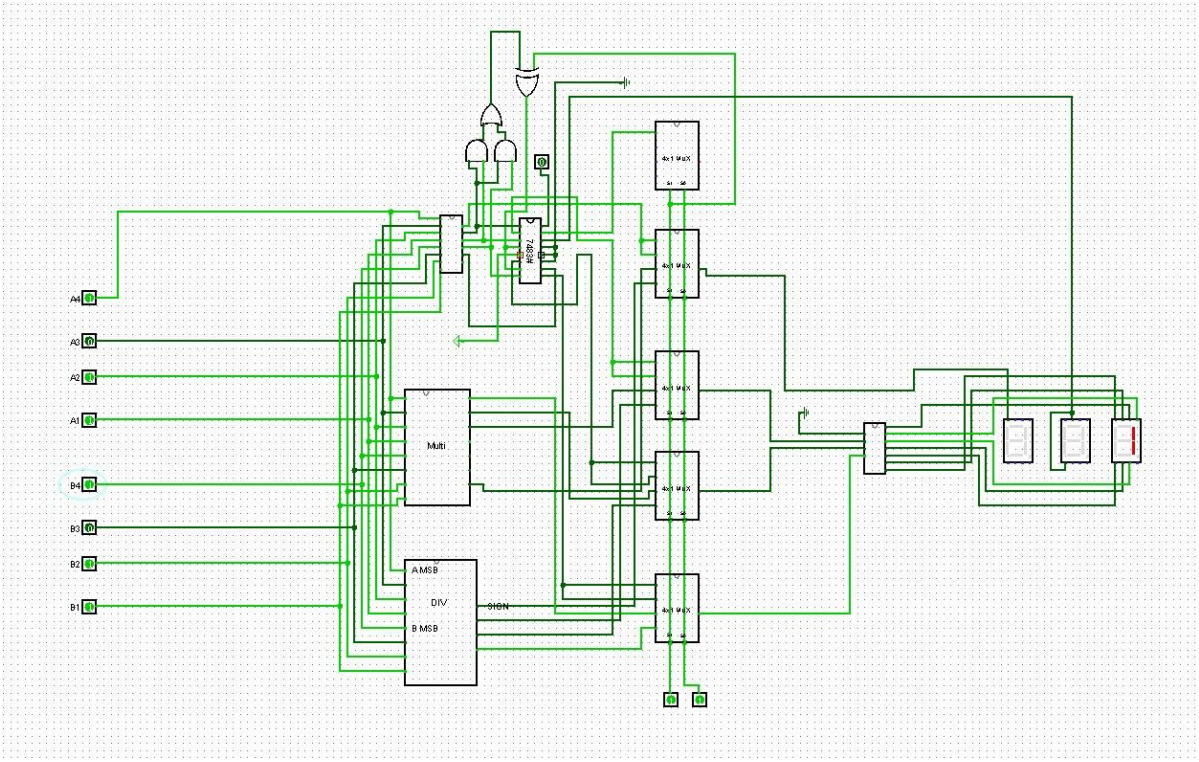
- Case (-7 / 2)



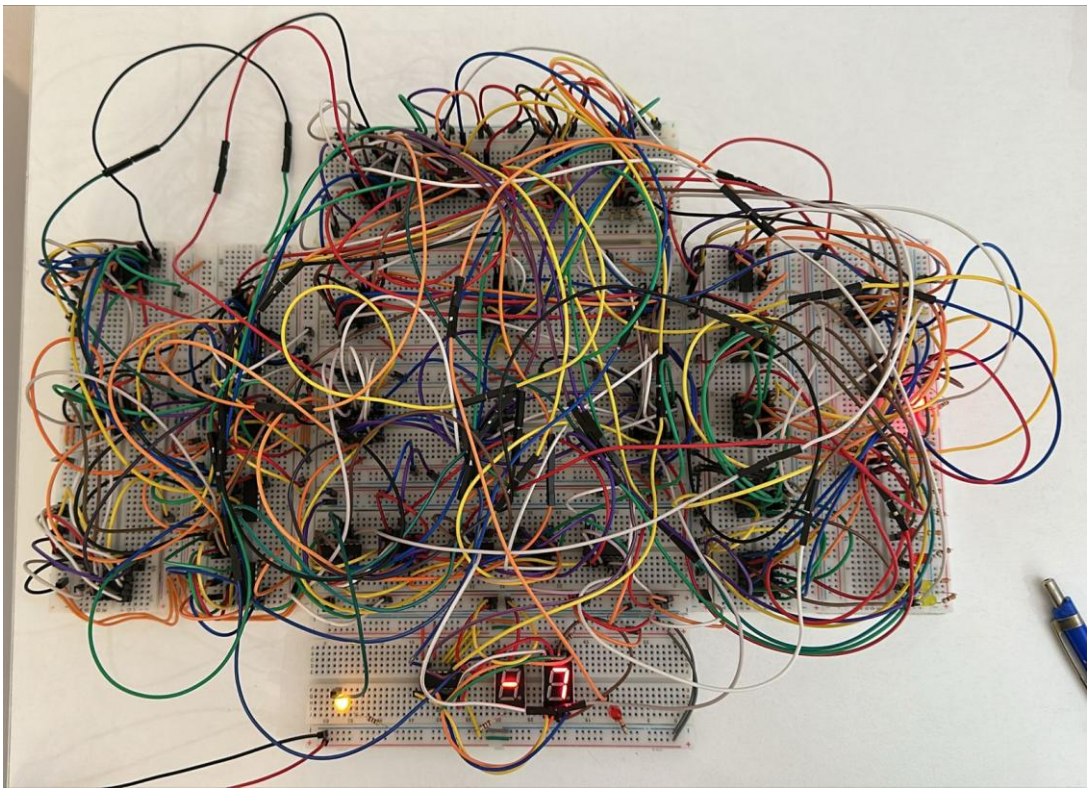


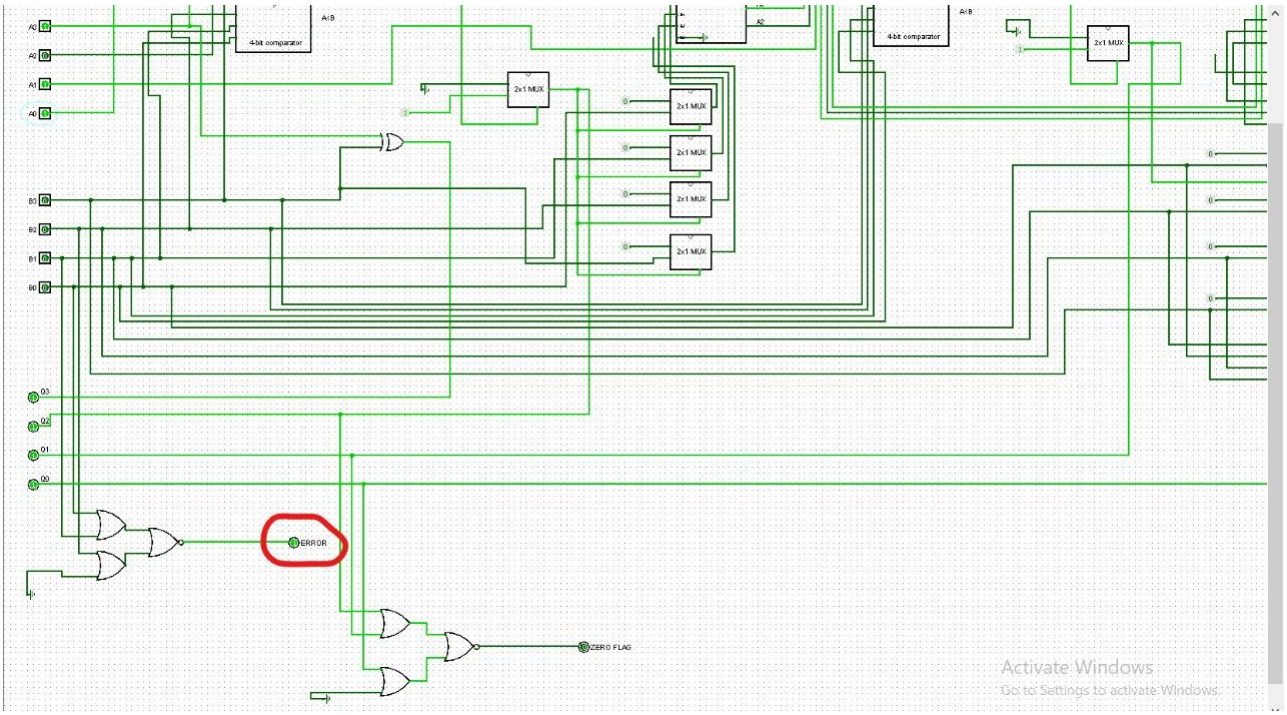
- Case (-3 / -3)



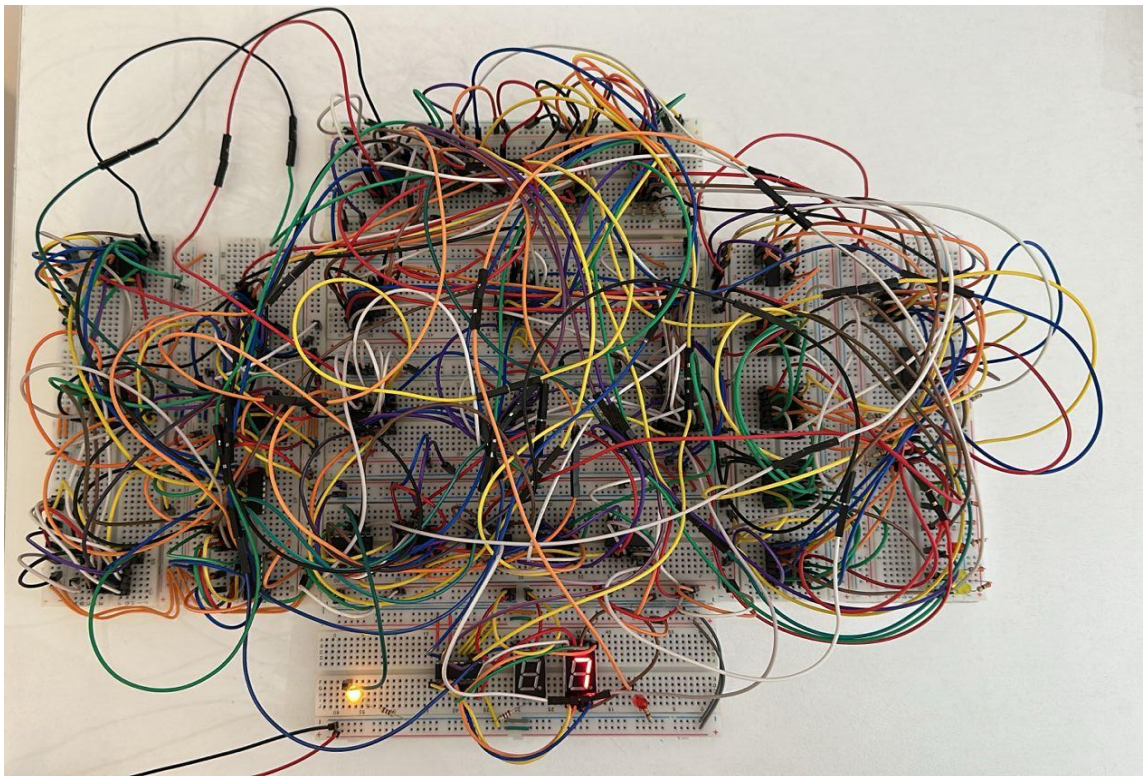

A4
A0
A2
A1
B4
B3
B2
B1
Multi
A MSB
DIV
B MSB
SIGN
4x1 MUX
4x1 MUX
4x1 MUX
4x1 MUX
4x1 MUX

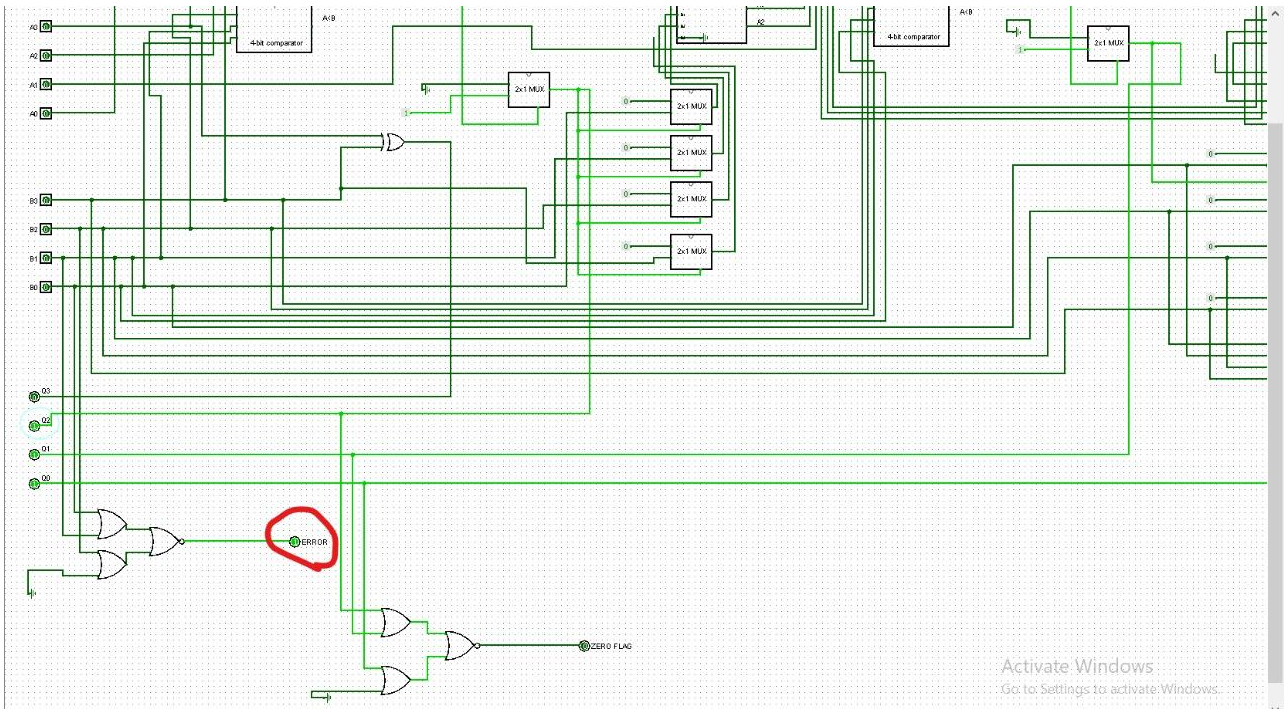
- Case (-3 / 0) Error yellow led



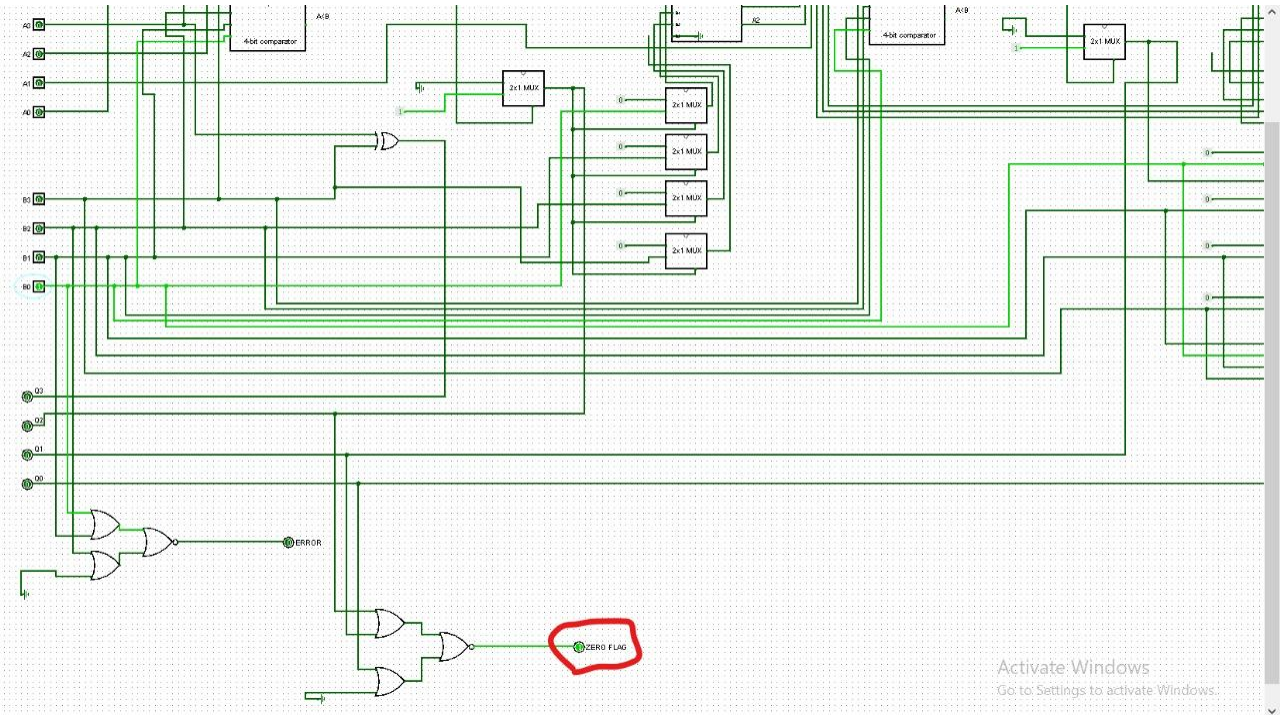


- Case (0 / 0) Error yellow led





- Zero Flag Led Case



7.0 Multiplexers

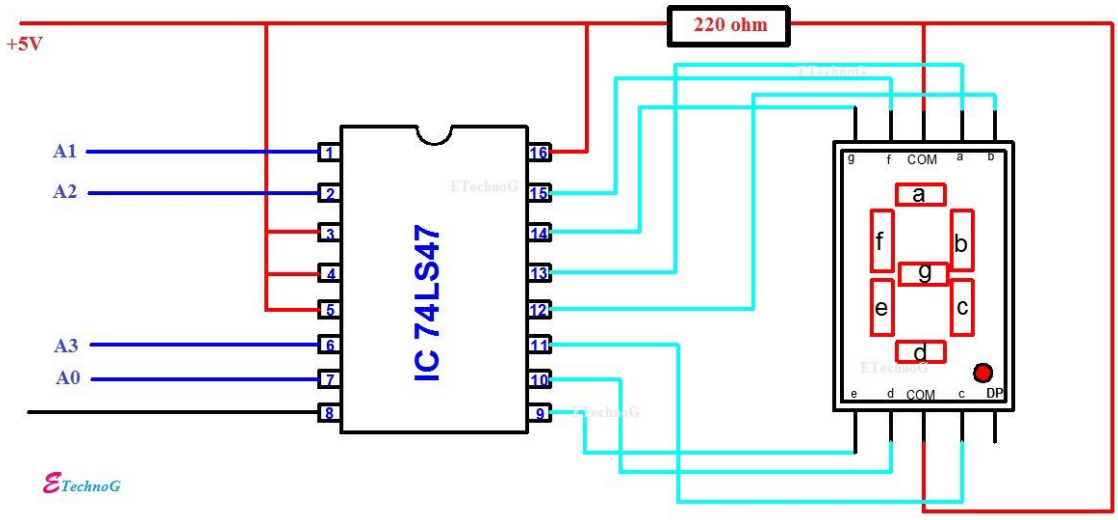
To control which arithmetic operation was performed in the ALU, we implemented a selection system using two 74153 dual 4-to-1 multiplexer ICs. The multiplexer receives inputs from all the arithmetic circuits—addition, subtraction, multiplication, and division—and passes only one of them to the output based on a 2-bit control signal. The control encoding was as follows:

- 00 → Addition
- 01 → Subtraction
- 10 → Multiplication
- 11 → Division

This configuration allowed us to select the active operation cleanly and efficiently, ensuring that the output always reflected the correct computation based on the selected mode.

8.0 7-Segment Displays and Decoder

For output display, we used two common anode 7-segment displays along with a 7447 BCD-to-7-segment decoder IC. The decoder handled the conversion from binary-coded decimal to the correct pattern of segments for visual display. One of the displays showed the absolute value (magnitude) of the result, while the second display indicated the sign (positive or negative) of the number. This setup allowed users to easily interpret the signed output values, especially since our hardware implementation was designed to operate within the range of -7 to +7.



9.0 Conclusion

In conclusion, this project demonstrated the design, simulation, and hardware implementation of a 4-bit Arithmetic Logic Unit capable of performing signed addition, subtraction, multiplication, and division. Each operation was carefully constructed using basic digital components such as multiplexers, full adders, and control logic, with dedicated circuits for overflow and error detection. Outputs were visualized through LEDs and seven-segment displays, with special circuits designed for BCD conversion where necessary. By combining both hardware and simulation, we were able to explore practical limitations and extend functionality for clearer visual output, ensuring both accurate operation and effective user feedback. This project provided comprehensive hands-on experience in digital system design and reinforced key concepts in arithmetic operations, logic control, and display interfacing.