Digital Logic Gate Simulation Exercise 2
Digital Adders and Subtractors
ELEE 2640

Benyamain Yacoob

January 25, 2024

Date Performed:	January 21, 2024
Partners:	Ara Oladipo
	Andre Price
Instructor:	Professor Paulik

Contents

1 Objectives

First Objective

Design, debug, and simulate, combinational circuits such as adder and subtractor circuits using MultiSIM SSI (small scale integration) and MSI (medium scale integration) 7400 series chips.

Second Objective

Investigate the operation of half adder, full adder, 2-bit full adder, and 4-bit full adder.

2 Materials

2-INPUT AND	74LS08N/7408N
2-INPUT OR	74LS32N
2-INPUT XOR	74LS386N/7486N
4-BIT FULL ADDER	74283N

3 Requirements

Half 1-Bit Adder

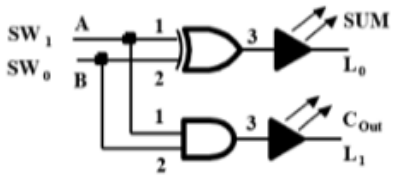

Figure 1: Half Adder Circuit Diagram

1-Bit Full Adder

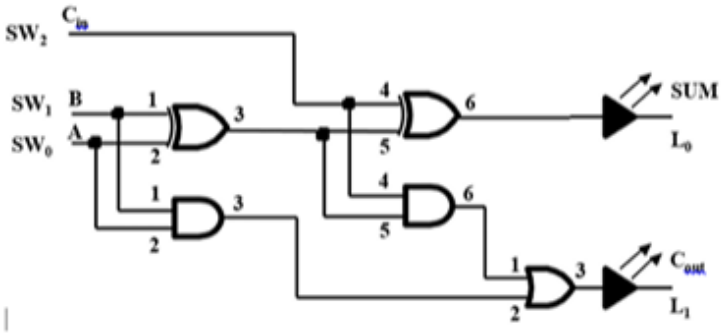

Figure 2: Circuit Diagram for 1-Bit Full Adder

4-Bit Full Adder

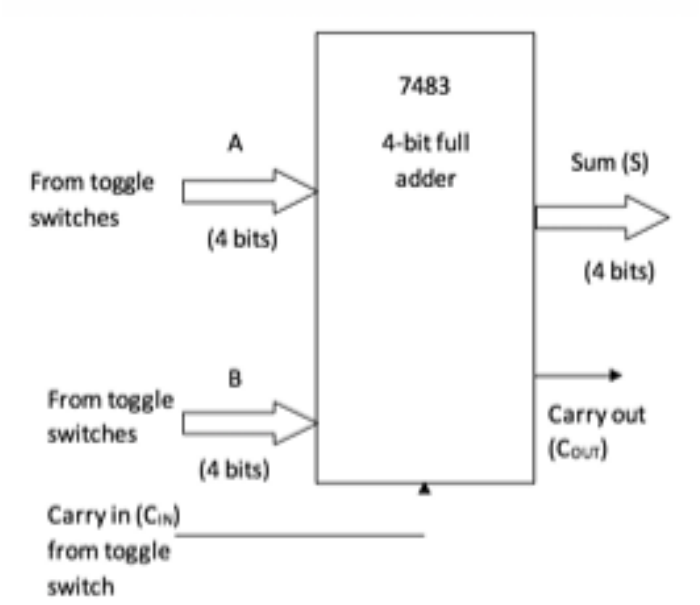

Figure 3: 4-Bit Full Adder Data Flow Diagram

4 Half 1-Bit Adder

4.1 Procedural Discussion and Experimental Setup

For us to create a half 1-bit adder, we started with using an “Interactive Digital Constant”. To get the chips and probes necessary to create the diagram, we went to

the “Components” menu from the “Place” options. The XOR Gate (7486N) and the AND Gate (7408N) were obtained from the TTL Group, and the LEDs were sourced from the Diodes Group. The wire connection was simple and we were able to easily create the SUM and C_{Out} outputs. We also tested the outputs by connecting the logic converter tools to the inputs, allowing us to create a truth table and using digital probes to see the output values as the simulation is running.

We were able to create a truth table using our current understanding of binary addition. We know that in binary addition when two 0s are added, the result will be 0. When a 0 is added to a 1, the result will be a 1 with no carry. Lastly, when two 1s are added, the result will be a 10, but the 0 will be placed in the same position and the 1 will carry over to the next bit. We will verify our truth table with the truth table outputted from the logic converter in Section 4.2.

Table 1: Truth Table for Half Adder

INPUTS		OUTPUTS	
A(SW_1)	B(SW_0)	Sum(L_0)	C_{out}(L_1)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$SUM = A'B + AB' \tag{1}$$

$$C_{out} = AB \tag{2}$$

Our circuit is shown below. The simulation is not running, but we will show the running circuit in Section **??**.

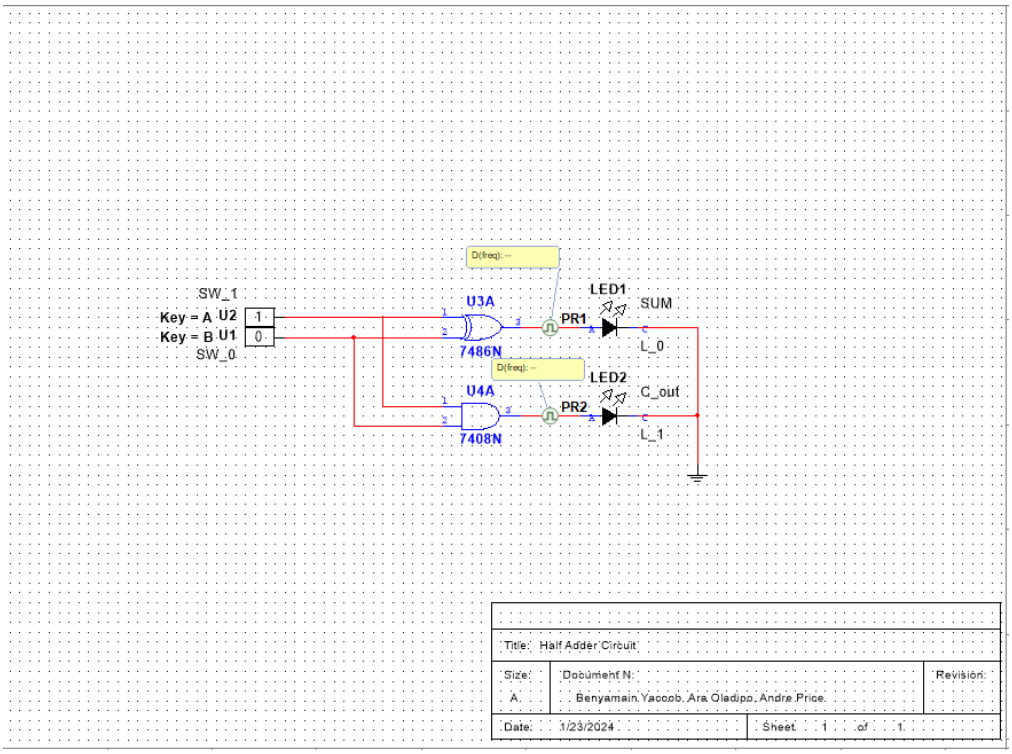

Figure 4: Half 1-Bit Adder Demonstration

4.2 Results and Analysis Discussion

Our functioning circuit is shown below. The first circuit shows a half 1-bit adder with one interactive digital constant on and the other off, receiving an input of 1 and 0. The SUM LED is on, showing that we are getting a proper output of 1. The second circuit image shows the circuit with both switches on. This turns on the CARRY LED, showing that the SUM output is 0, and the CARRY is 1.

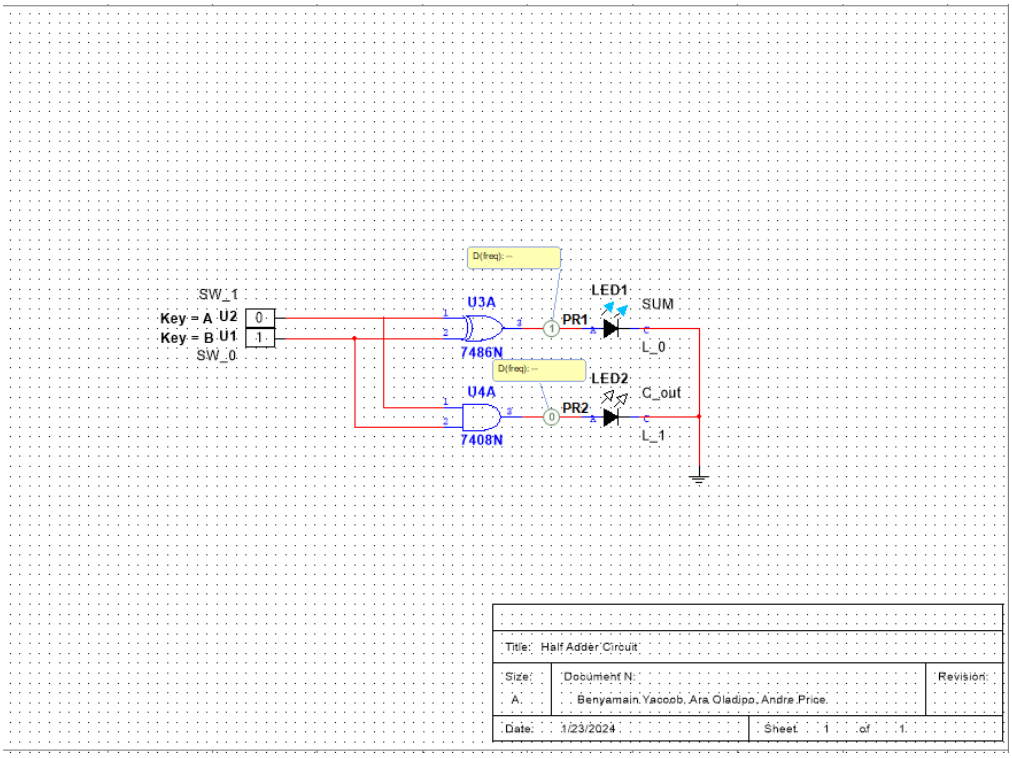

Figure 5: Half 1-Bit Adder (0 + 1)

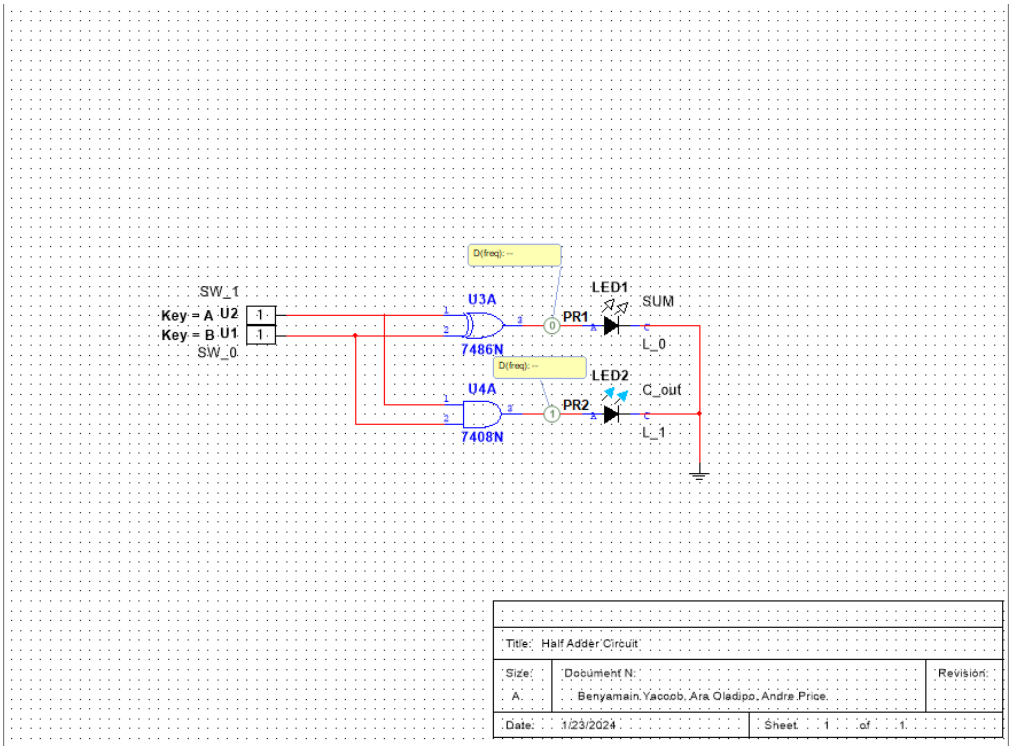

Figure 6: Half 1-Bit Adder (1 + 1)

Furthermore, we verified our circuit with the MultiSIM word generator tool. We initially faced some issues with understanding how to set up the word generator, and we were confused about how we would connect the inputs. Eventually, after watching a few YouTube videos, we were able to make progress and get it to work. The functioning word generator connected to our half 1-bit adder is shown below.

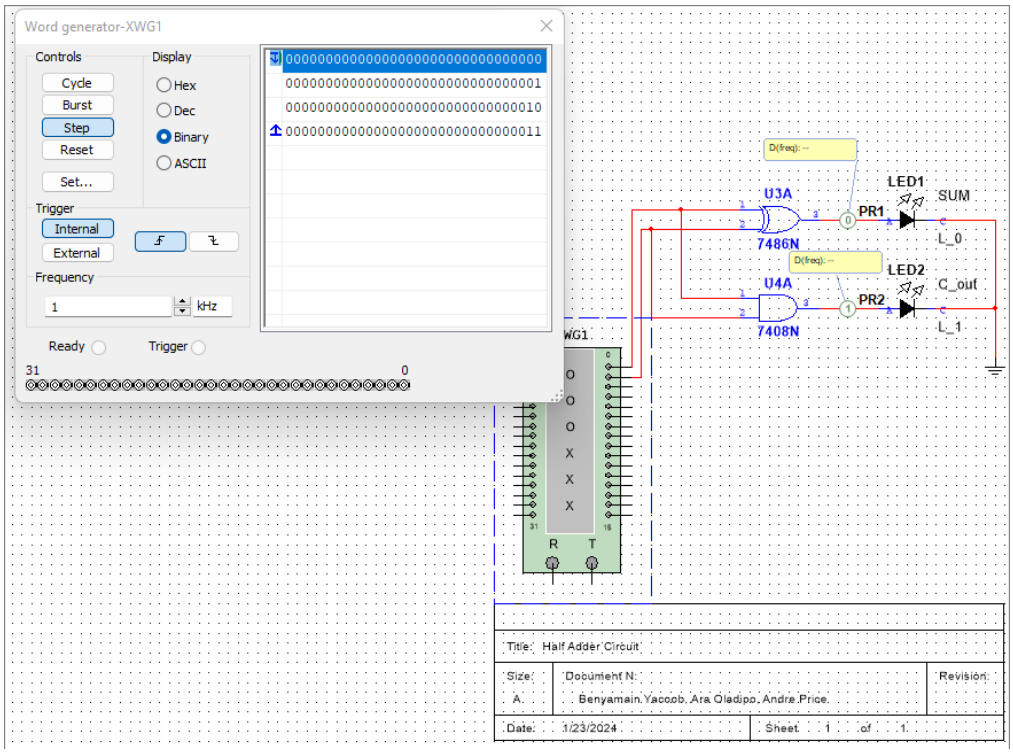

Figure 7: Half Adder Word Generator 1

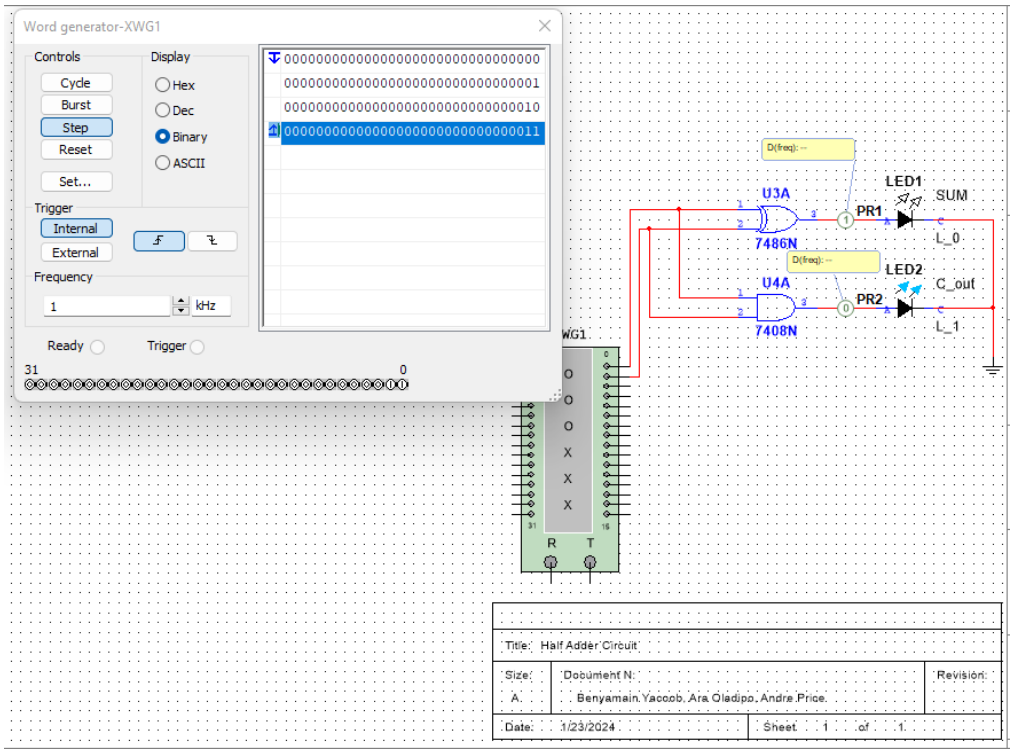

Figure 8: Half Adder Word Generator 2

Our circuit shows that when both digital constants are off, the SUM and the CARRY output will be 0. When one of the switches is on, the SUM output will be 1, while the CARRY output will be 0, and when both switches are on, the SUM and CARRY output will be 1. This verifies what is shown in our truth table, but we will be using a logic converter to verify this.

The truth tables generated by MultiSIM's logic converter for the SUM and C_{out} output are shown below:

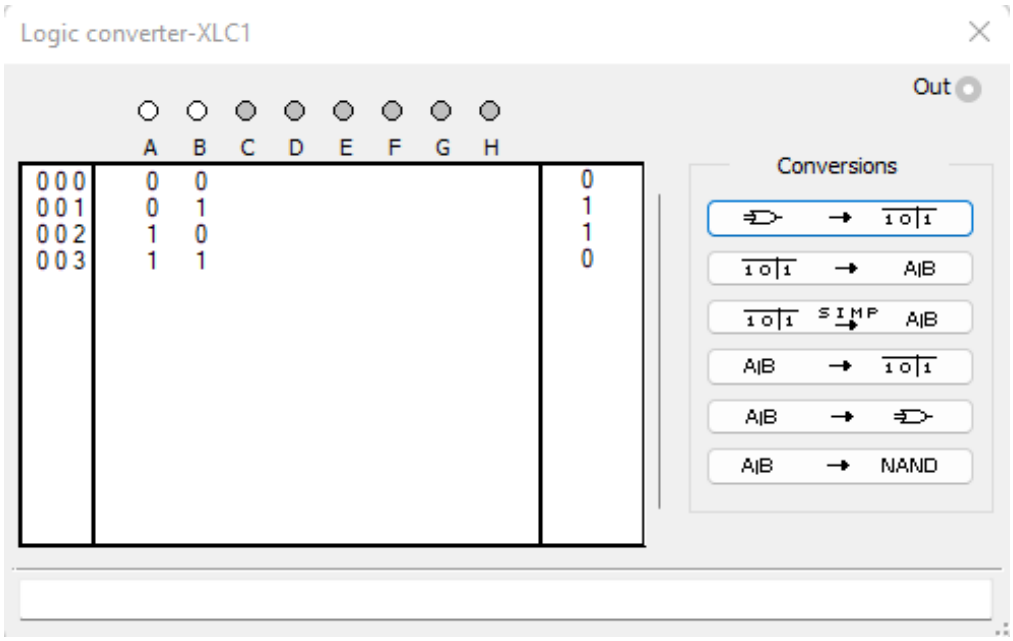

Figure 9: Half 1-Bit Adder SUM Truth Table

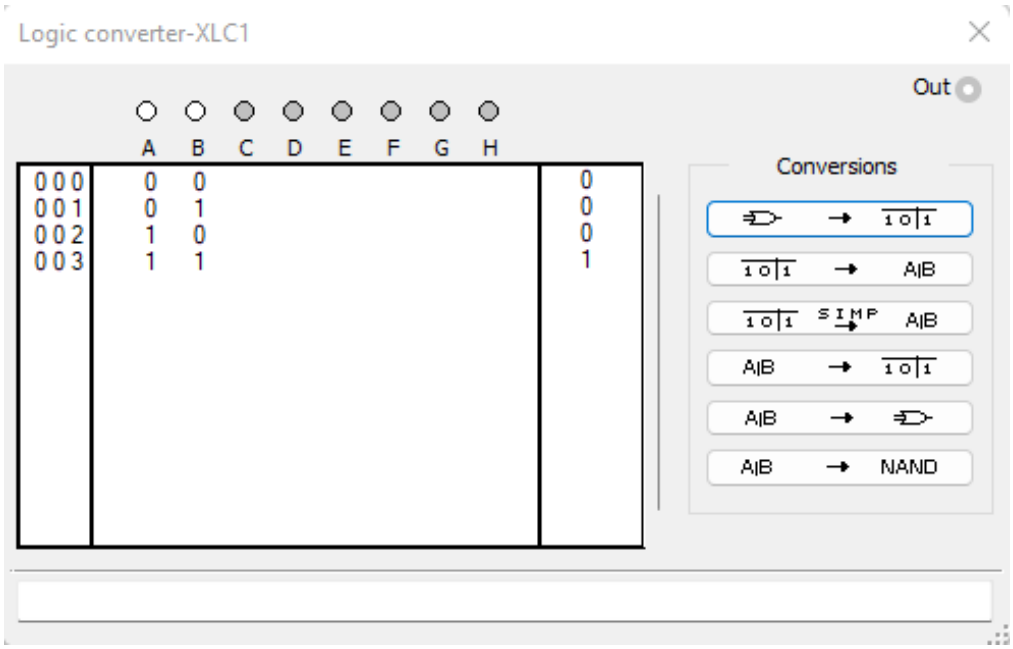

Figure 10: Half 1-Bit Adder CARRY Truth Table

This shows that there are no discrepancies between our theory and the results, as our truth table matches the truth table generated by the MultiSIM logic converter. The process of creating the half 1-bit adder was straightforward, and we were able to complete it relatively quickly. Essentially our experiments confirmed that a half adder is a circuit whose output is the binary sum of its two 1-bit inputs, with the SUM output producing the result of the addition, while the CARRY output indicates the carry-over from the addition of the two inputs.

5 1-Bit Full Adder

5.1 Procedural Discussion and Experimental Setup

Concerning the theory of the 1-bit full adder, we will have three inputs, A, B, and C_{in}, and two outputs, SUM and C_{out}. Inputs A and B will represent 1-bit binary numbers, while the C_{in} signal (along with the C_{out} signal) will be used when adding numbers that are more than one bit too long. Just as in the half-bit adder section, using our knowledge of full adders, we will be creating a truth table, then using MultiSIM to verify this theory, and using our results and analysis to reconcile any discrepancies.

We know that a full adder functions similarly to the half adder, but it has a CARRY input (C_{in}). This allows us to create the truth table by simulating a binary addition with a CARRY input (when C_{in} is 1). When the CARRY input is 0, the output of the full adder ($C_{out}(L_1)$ and Sum(L_0)) should be the same as that out a half adder. The truth table theory, along with the simplified logic function is shown below:

Table 2: Truth Table for 1-Bit Full Adder

INPUTS			OUTPUTS	
$C_{in}(SW_2)$	B(SW_1)	A(SW_0)	Sum(L_0)	$C_{out}(L_1)$
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

$$SUM = A'B'C + A'BC' + AB'C' + ABC \tag{3}$$

$$C_{out} = AC + AB + BC \tag{4}$$

The process to make the circuits for the 1-bit full adder was very similar to the process for the half-bit adder, but it made use of 2 XOR gates, 2 AND gates, and 1 OR gate. The demonstration of our circuit is shown below. The working circuits are shown in Section 5.2. This circuit was also straightforward to create, and we did not face any problems making it.

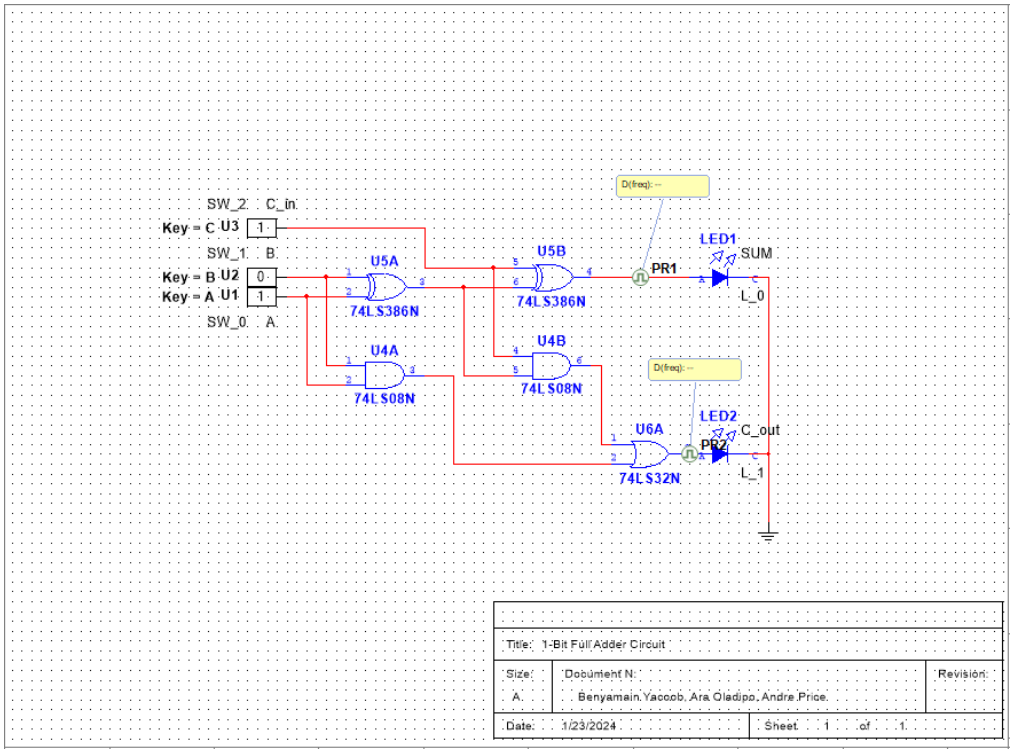

Figure 11: 1-Bit Full Adder

5.2 Results and Analysis Discussion

Our functioning circuits are shown below. The first circuit shows the circuit with a CARRY bit of 1, along with two input bits of 1. If this was a half-bit adder, there would be no C_{in}, so if we had two input bits of 1, then the SUM output would be 0 and the carry output would be 1. Since this is a full adder and we have a CARRY input bit of 1, then we would have 1 for the SUM output and the C_{out} output.

The second circuit image shows the circuit when the CARRY bit is 1, and there is only 1 input bit — the second input bit is 0. This will provide a SUM output of 0 and a CARRY output of 1. This also confirms what we had in our truth table.

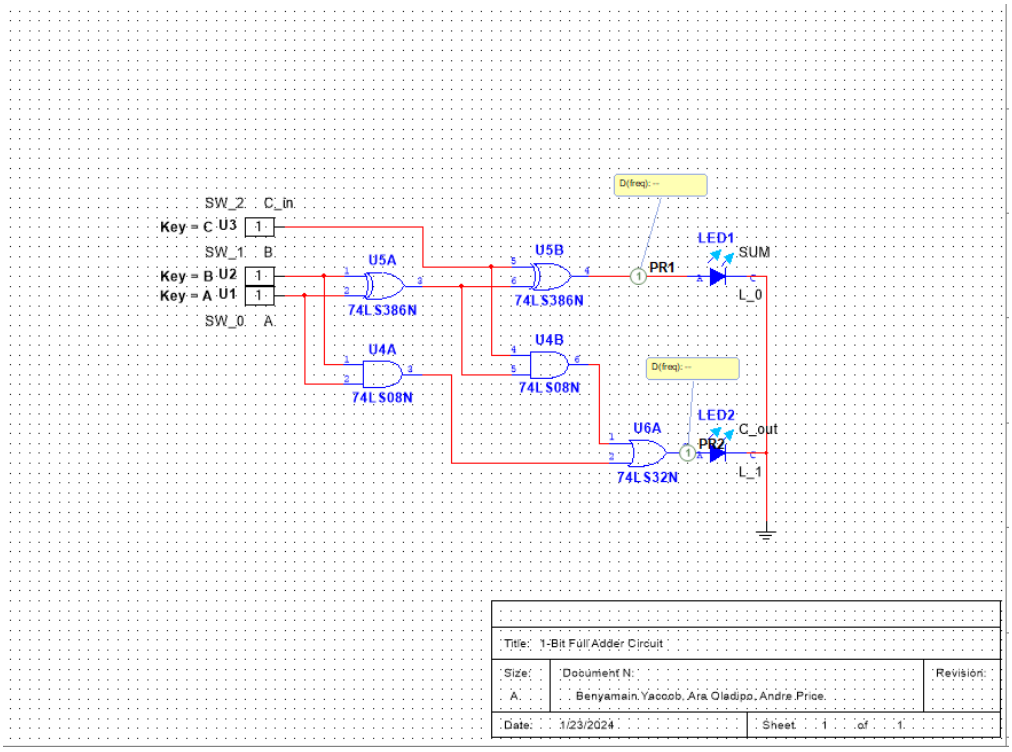

Figure 12: 1-Bit Full Adder 1 CARRY Bit With (1 + 1)

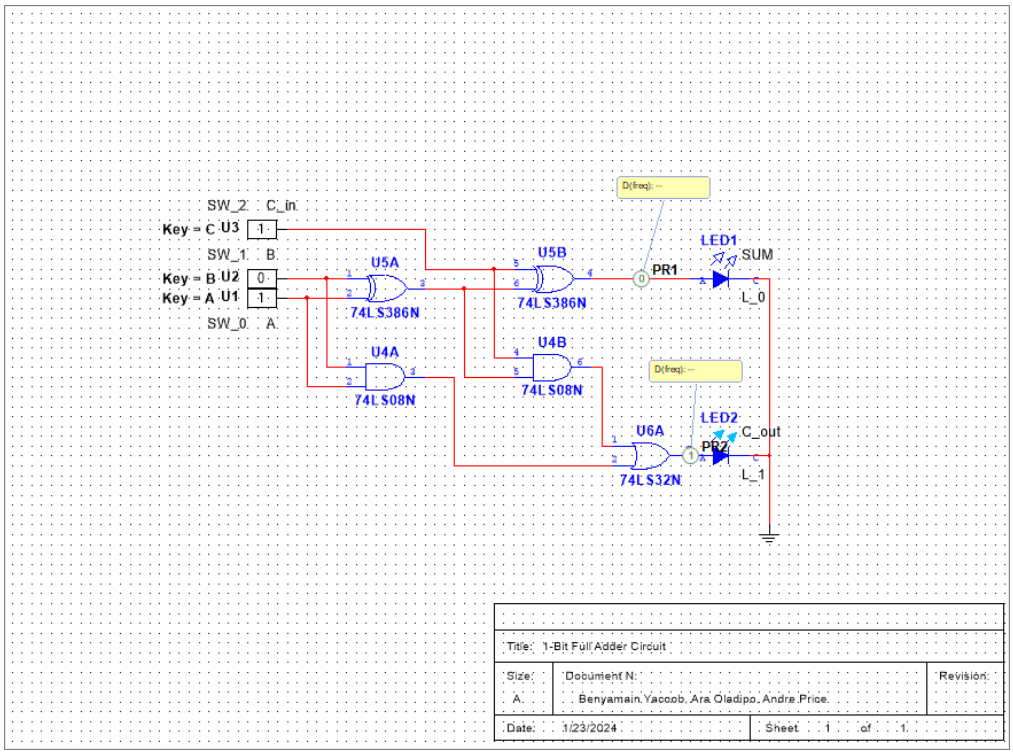

Figure 13: 1-Bit Full Adder 1 CARRY Bit With (0 + 1)

The truth table provided by the logic converter is shown below.

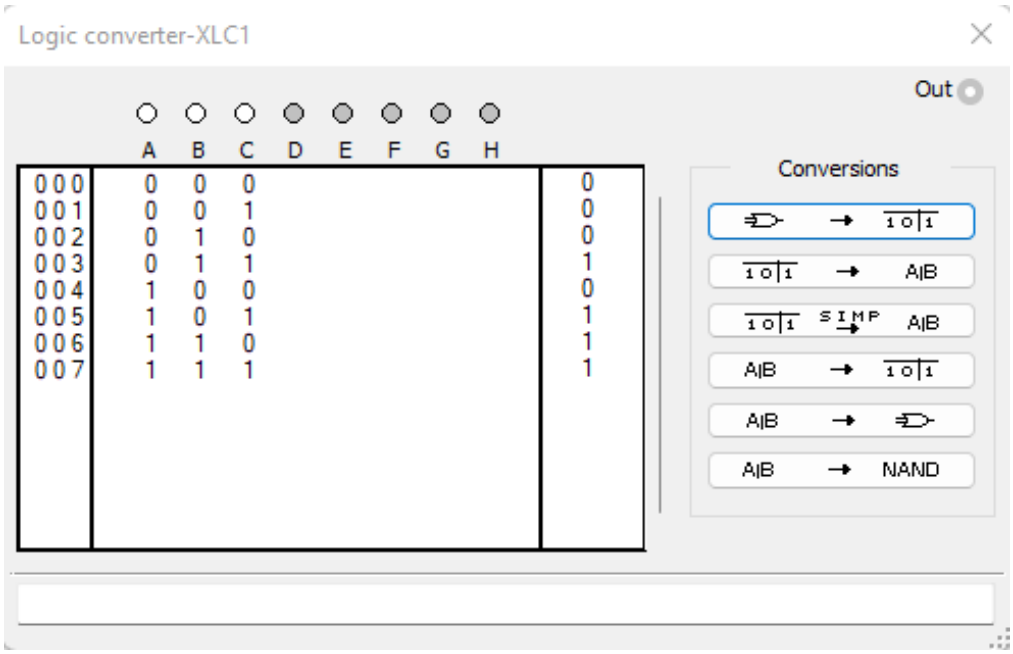

Figure 14: 1-Bit Full Adder CARRY Truth Table

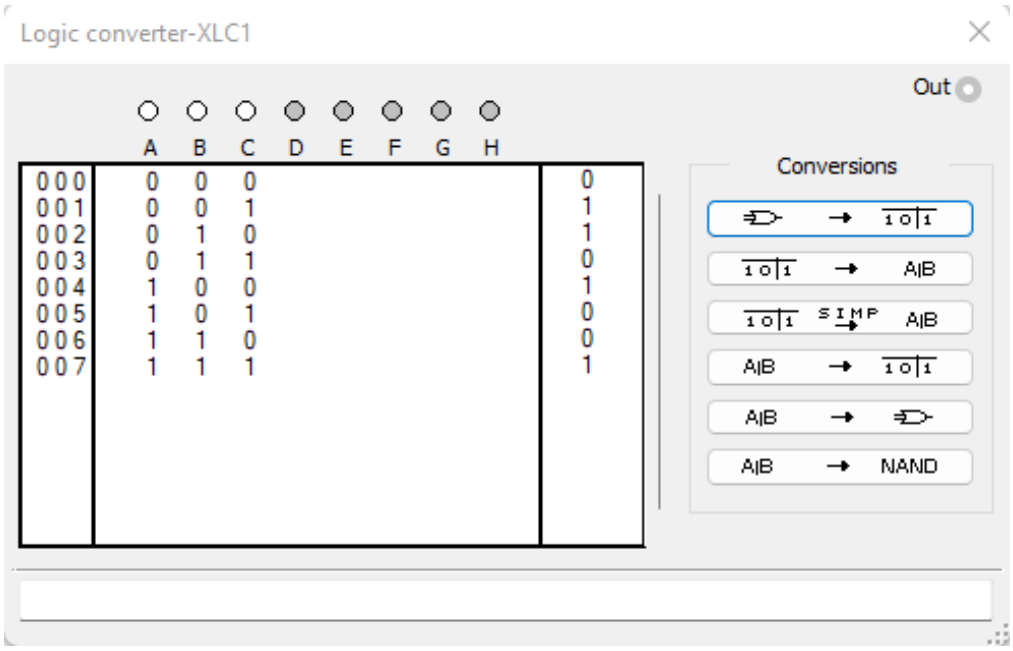

Figure 15: 1-Bit Full Adder SUM Truth Table

This shows that there were no discrepancies between our theory truth table and the truth table generated by MultiSIM. Finally, to exercise our circuit, we also used a word generator. After our initial struggles with using the tool, we found that it was easier to set it up for this circuit, and we were able to get the desired results.

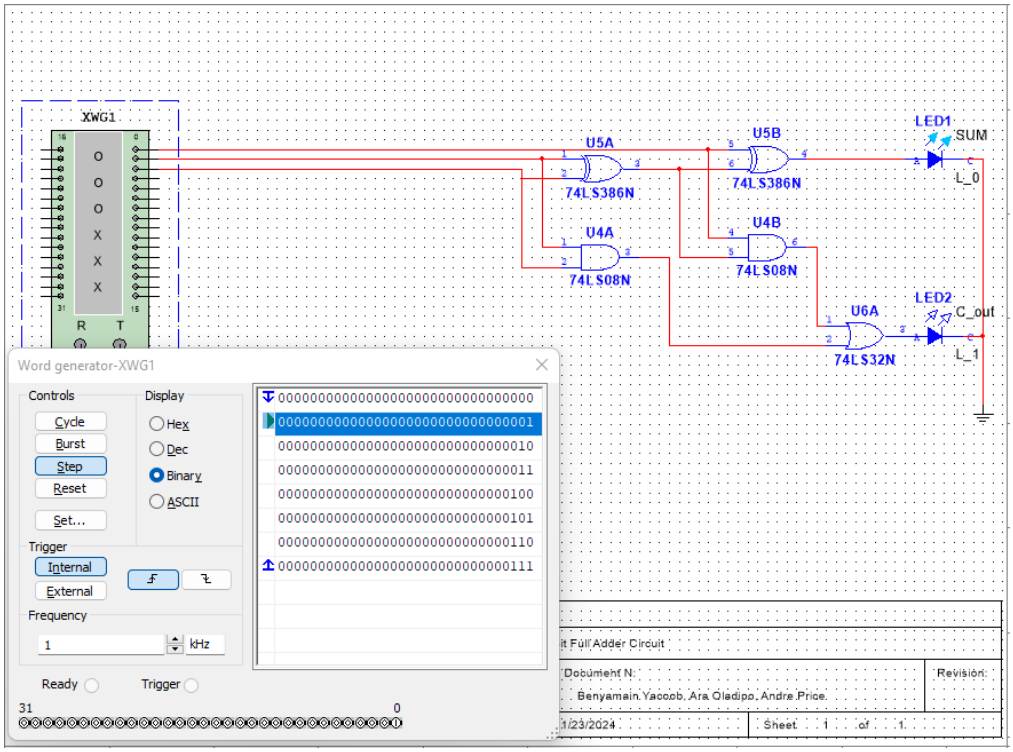

Figure 16: Full Adder Word Generator 1

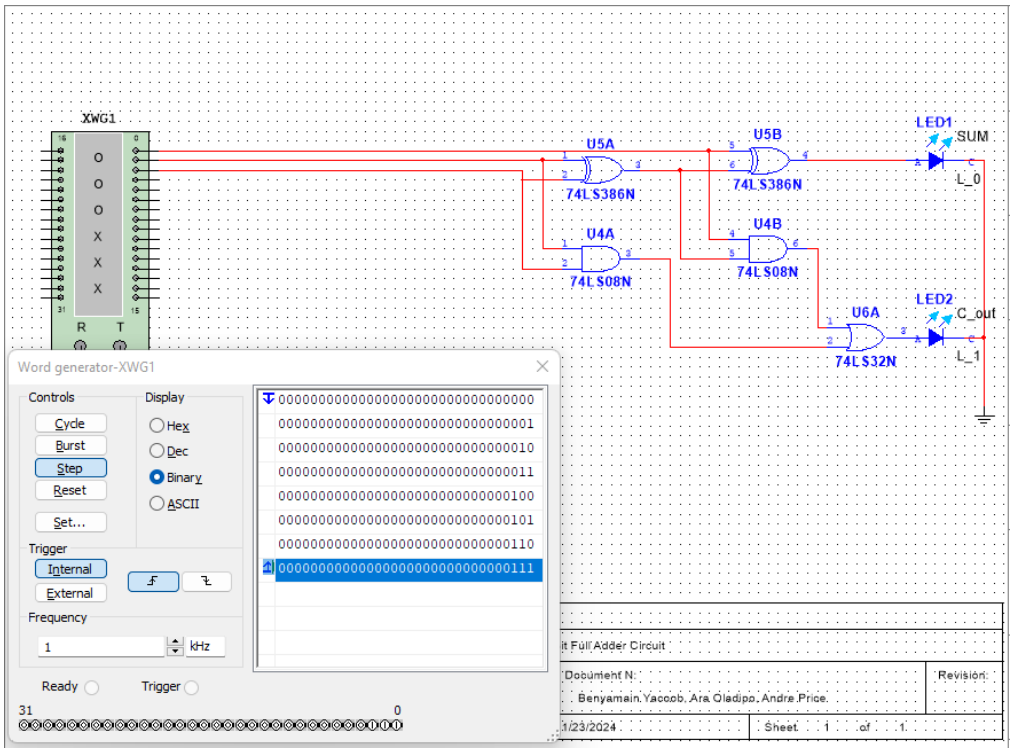

Figure 17: Full Adder Word Generator 2

6 4-Bit Binary Adder

6.1 Procedural Discussion and Experimental Setup

A 4-bit binary adder can be constructed with four 1-bit full adders, with the output carry from each full adder connected to the input carry from the next full adder. It essentially simulates the binary addition process with multiple bits. Following the guidance of the assignment rubric, we can have the first 4-bit number, A, represented as $A_4A_3A_2A_1$, while the second 4-bit number is $B_4B_3B_2B_1$. The 4-bit sum of A and B will be represented as $SUM_4SUM_3SUM_2SUM_1$, with C_in being the carry input and C_out being the carry output.

To create our circuit, we used interactive digital constants as an alternative to switches and a power source. We sourced the "4-Bit Full Adder With Fast Carry" from the TTL Group (74283N). We also used a 4-bit HEX display ("DCD_HEX_DISPLAY") to monitor our outputs, along with a word generator. Lastly, we used a digital probe to verify the CARRY output. Our generated truth table is shown below:

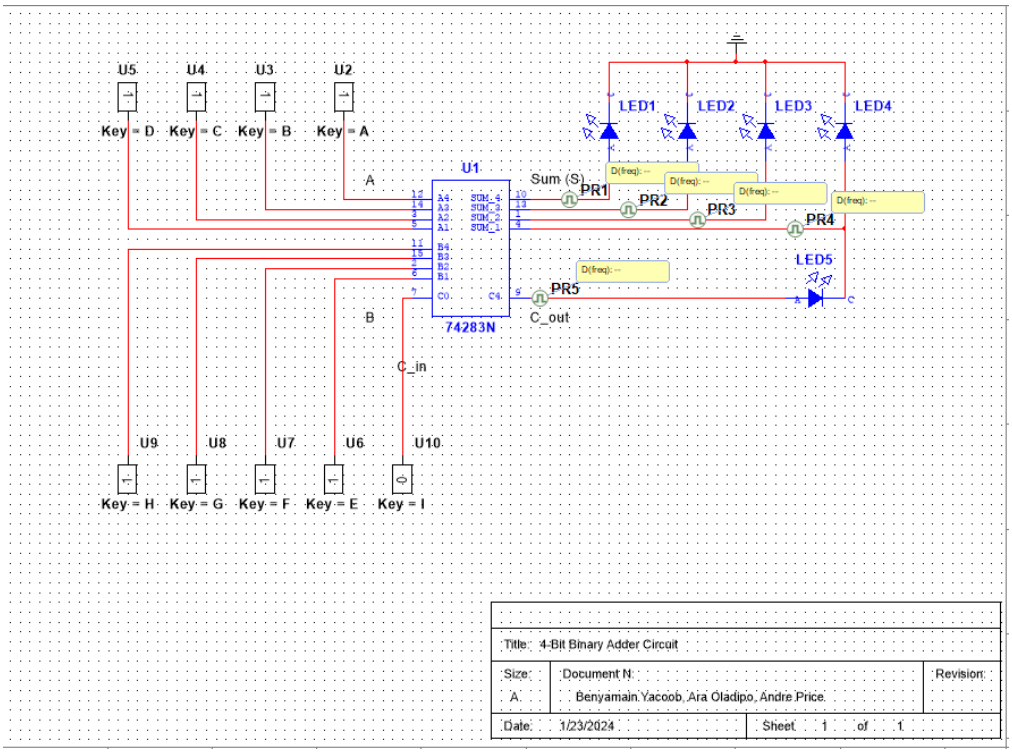

Figure 18: 4-Bit Full Adder

By hand, we calculated what the truth table of a 4-bit binary adder would look like, and following the assignment rubric, the output will be displayed as HEX digits. We will verify this generated truth table in Section 6.2. The theory truth table is shown below:

Truth Table for 4-bit Adder: HEX outputs.

$M = 0$

INPUTS								CALC OUT		ACTUAL OUT		Check if Correct
A_4	A_3	A_2	A_1	B_4	B_3	B_2	B_1	C_{out}	S_{out}	C_{out}	S_{out}	✓
0	1	0	1	0	1	0	1	0	A			
0	1	1	1	0	1	0	1	0	C			
1	0	0	1	0	1	0	1	0	D			
1	0	1	1	0	1	0	1	1	F			

$$SUM_1 = A'B'C'D'E'H + A'B'C'F'H + A'B'D'E'F'H + A'CFGH' + A'CDGH' + A'DFGH' + A'CEGH' + A'B'G'H + A'EFGH' + A'C'D'E'G'H + A'C'F'G'H + A'D'E'F'G'H + A'BCFH' + A'BCDH' + A'BDFH' + A'BCEH' + A'BEFH' + A'BGH' + AB'C'D'E'H' + AB'C'F'H' + AB'D'E'F'H' + ACFGH + ACDGH + ADFGH + ACEGH + AB'G'H' + AEFGH + AC'D'E'G'H' + AC'F'G'H' + AD'E'F'G'H' + ABCFH + ABCDH + ABDFH + ABCEH + ABEFH + ABGH \quad (5)$$

$$SUM_2 = B'C'D'E'G + B'C'F'G + B'D'E'F'G + B'CFG' + B'CDG' + B'DFG' + B'CEG' + B'EFG' + BC'D'E'G' + BC'F'G' + BD'E'F'G' + BCFG + BCDG + BDFG + BCEG + BEFG \quad (6)$$

$$SUM_3 = C'D'E'F + C'DF' + C'EF' + CD'E'F' + CDF + CEF \quad (7)$$

$$SUM_4 = D'E' + DE \quad (8)$$

$$C_{out} = CFGH + CDGH + DFGH + CEGH + EFGH + BCFH + BCDH + BDFH + BCEH + BEFH + BGH + ACFG + ACDG + ADFG + ACEG + AEFG + ABCF + ABCD + ABDF + ABCE + ABEF + ABG + AH \quad (9)$$

6.2 Results and Analysis Discussion

The truth table generated by the logic converter confirmed our results. Since the logic converter only has one output, we had to generate four truth tables per output to verify our result. The final truth table is shown below:

Truth Table for 4-bit Adder: HEX outputs.

$M = 0$

INPUTS								CALC OUT		ACTUAL OUT		Check if Correct
A_4	A_3	A_2	A_1	B_4	B_3	B_2	B_1	C_{out}	S_{out}	C_{out}	S_{out}	✓
0	1	0	1	0	1	0	1	0	A	0	A	✓
0	1	1	1	0	1	0	1	0	C	0	C	✓
1	0	0	1	0	1	0	1	0	D	0	D	✓
1	0	1	1	0	1	0	1	1	F	1	F	✓

The circuit images are shown below:

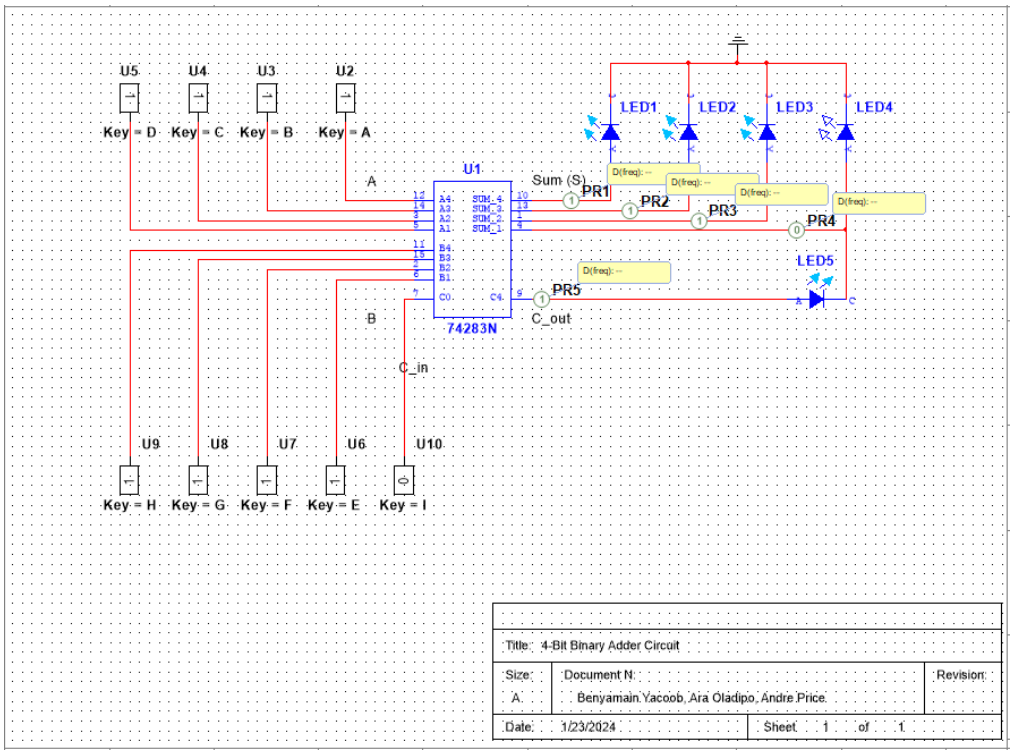

Figure 19: 4-Bit Binary Adder Demonstration 1

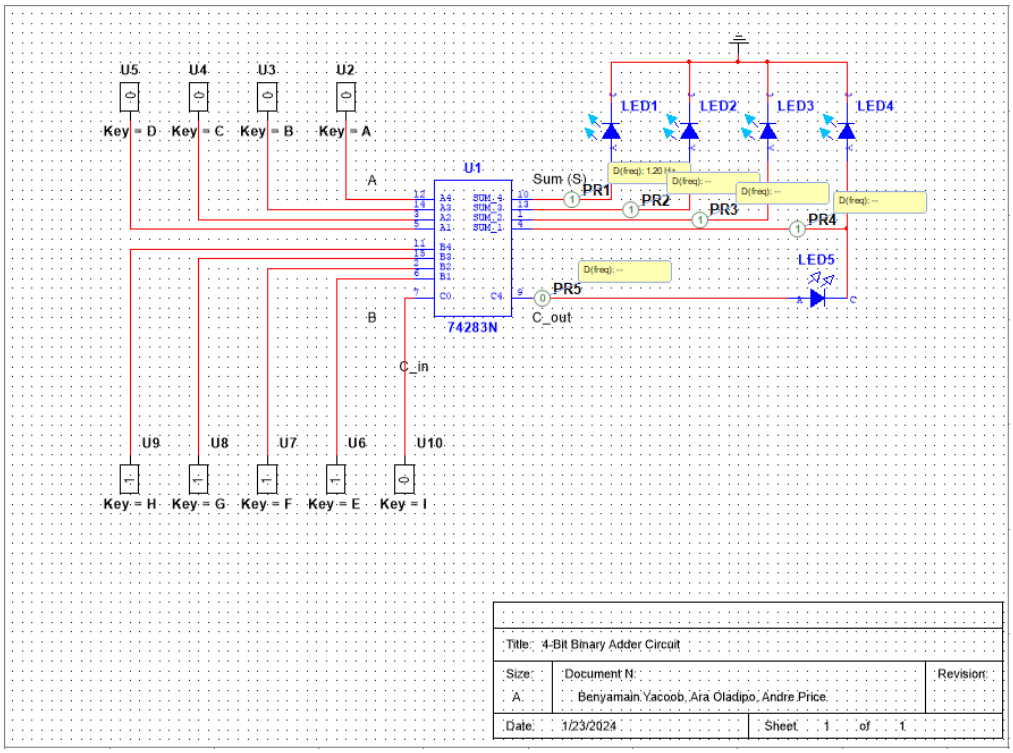

Figure 20: 4-Bit Binary Adder Demonstration 2

We also verified our circuits using a word generator with HEX output. The results are shown below:

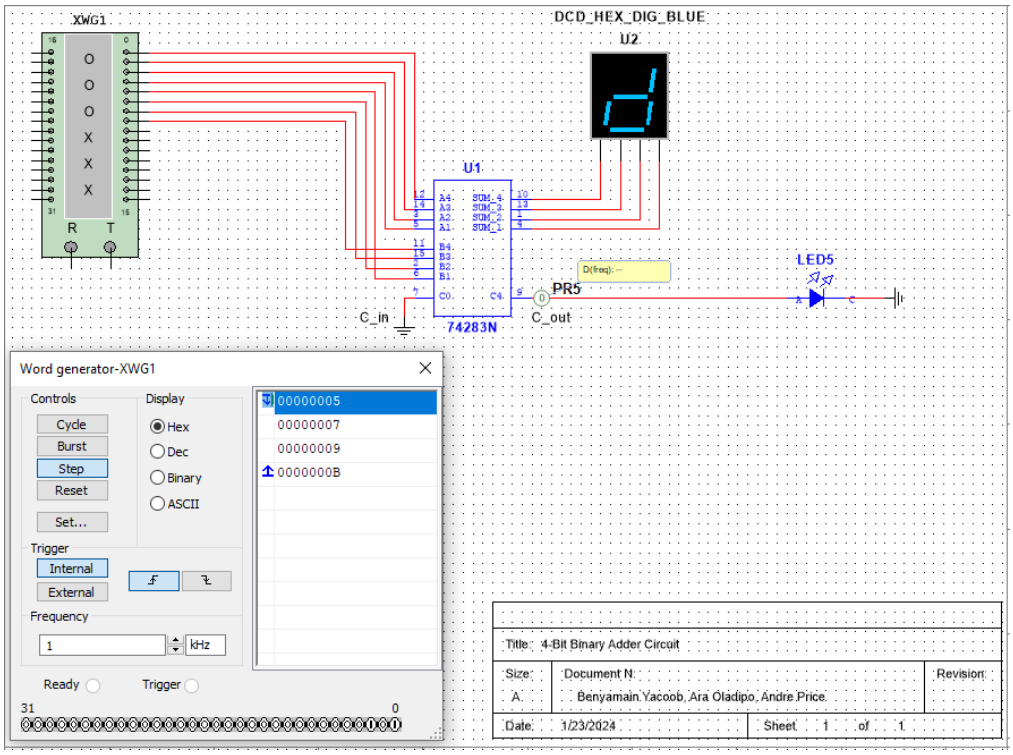

Figure 21: 4-Bit Binary Adder Word Generator A-Input Sequence 1

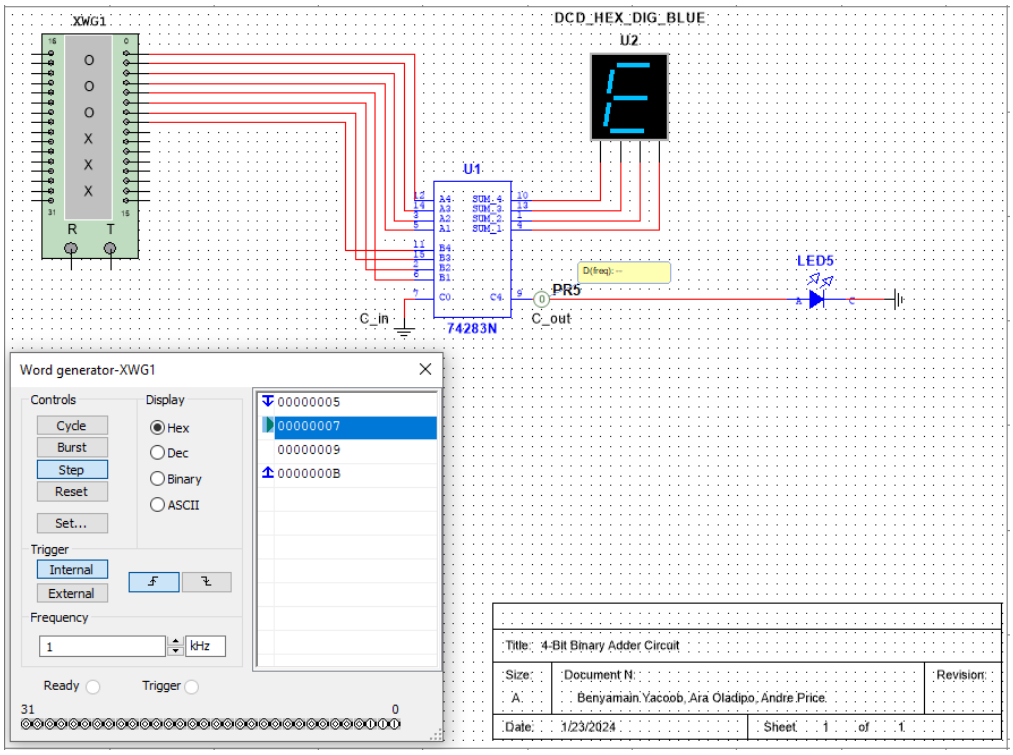

Figure 22: 4-Bit Binary Adder Word Generator A-Input Sequence 2

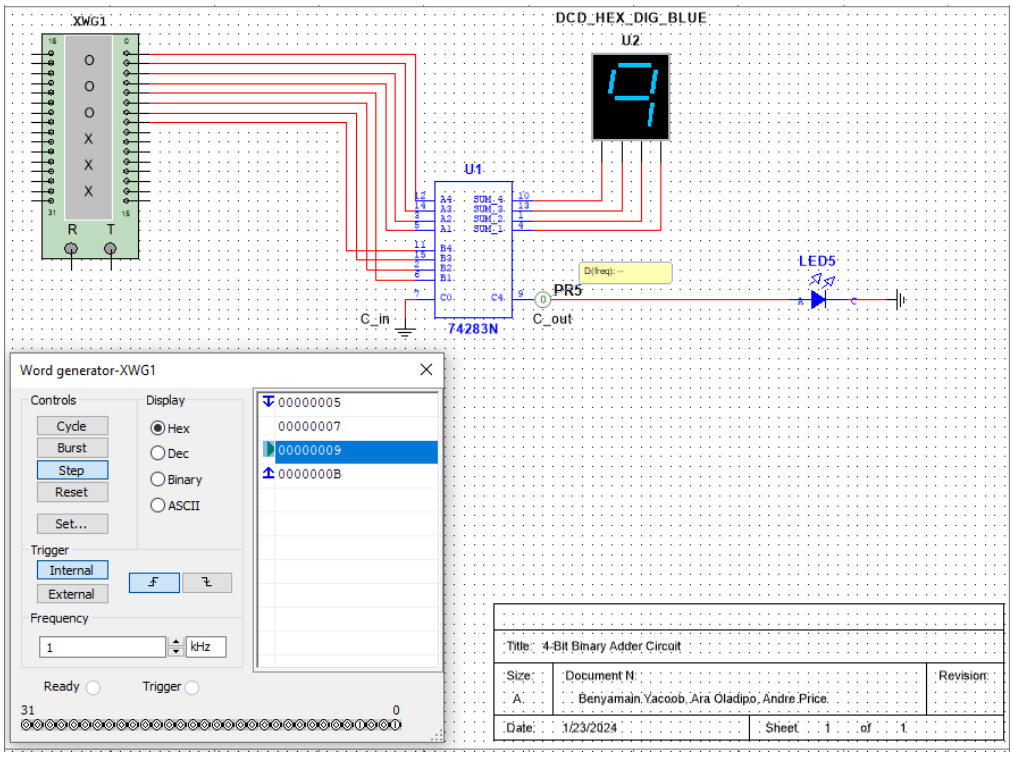

Figure 23: 4-Bit Binary Adder Word Generator A-Input Sequence 3

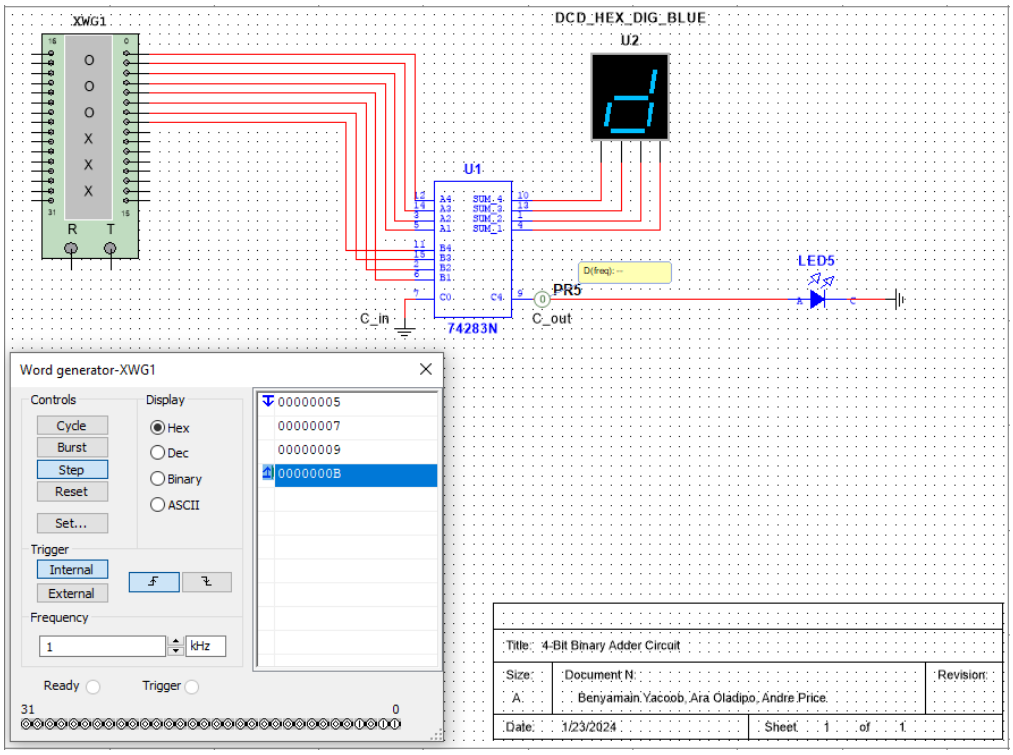

Figure 24: 4-Bit Binary Adder Word Generator A-Input Sequence 4

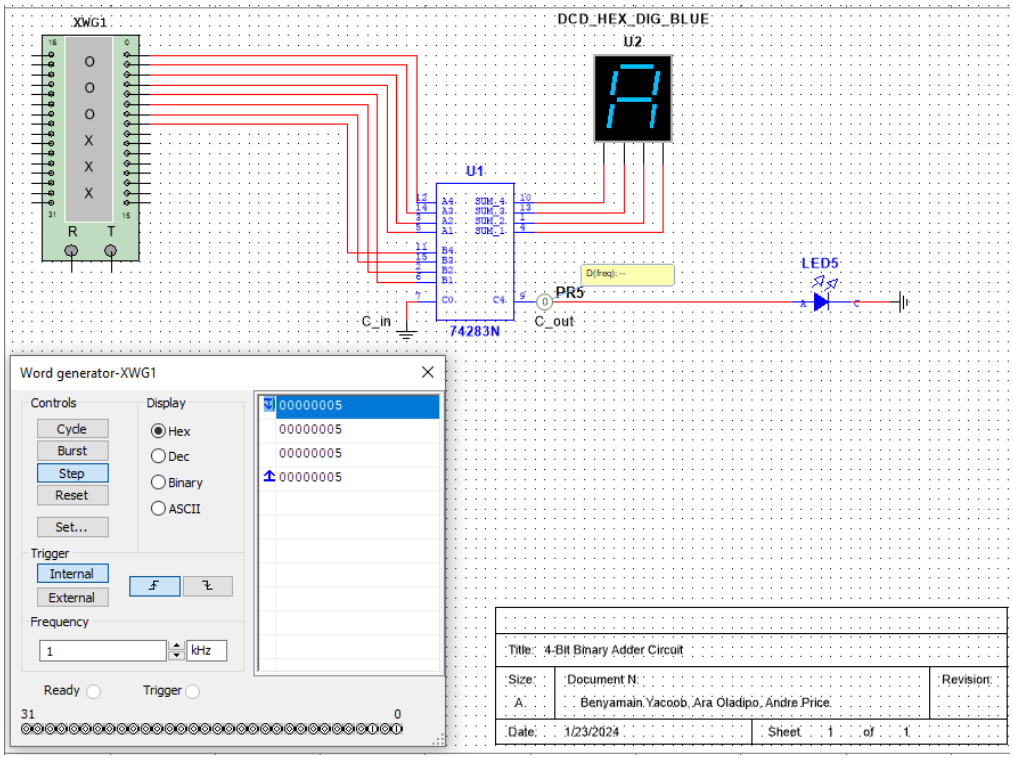

Figure 25: 4-Bit Binary Adder Word Generator B-Input Sequence

7 4-Bit Binary Subtractor

A 4-bit binary subtractor is similar to a 4-bit binary adder, but instead of mode M being 0, M will be 1. This means that the initial carry (C_0) will be 1. Since we want to subtract, we will add the two's complement of B to A, which gives us the same result as subtraction. In practice, this shows as adding A to the 1's complement of B, but with a carry input of 1. The 4-bit addition of A and the two's complement of B will be represented as $SUM_4SUM_3SUM_2SUM_1$, with C_{in} being the carry input of 1, and C_{out} being the carry output.

7.1 Procedural Discussion and Experimental Setup

Following the rubric of the assignment, we used our understanding of 4-bit subtractor to partially complete the provided truth table. The partially completed truth table is shown below:

Truth Table for 4-bit Subtractor: HEX outputs.

$M = 1$												
INPUTS								CALC OUT		ACTUAL OUT		Check if Correct
A_4	A_3	A_2	A_1	B_4	B_3	B_2	B_1	C_{out}	S_{out}	C_{out}	S_{out}	✓
0	1	0	1	0	1	0	1	0	0			
0	1	1	1	0	1	0	1	0	2			
1	0	0	1	0	1	0	1	0	4			
1	0	1	1	0	1	0	1	0	6			

The process of creating the circuit was tedious, and we came across some issues, but we were able to resolve them quickly. An image of our circuits is shown below:

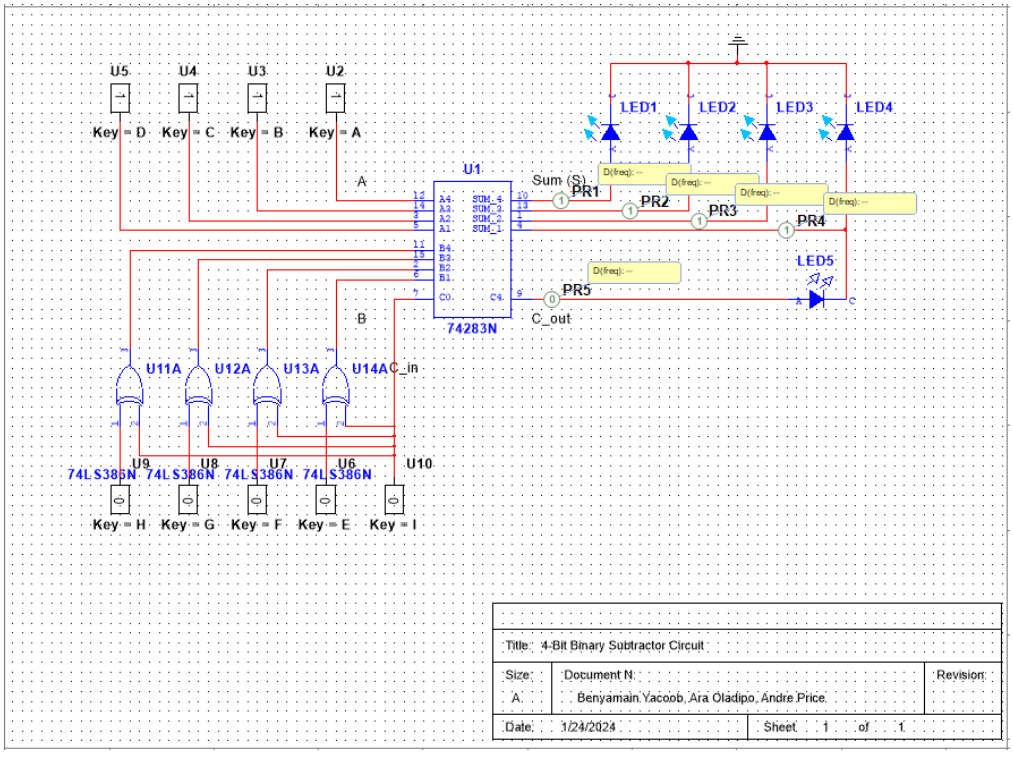

Figure 26: 4-Bit Binary Subtractor Demonstration 1

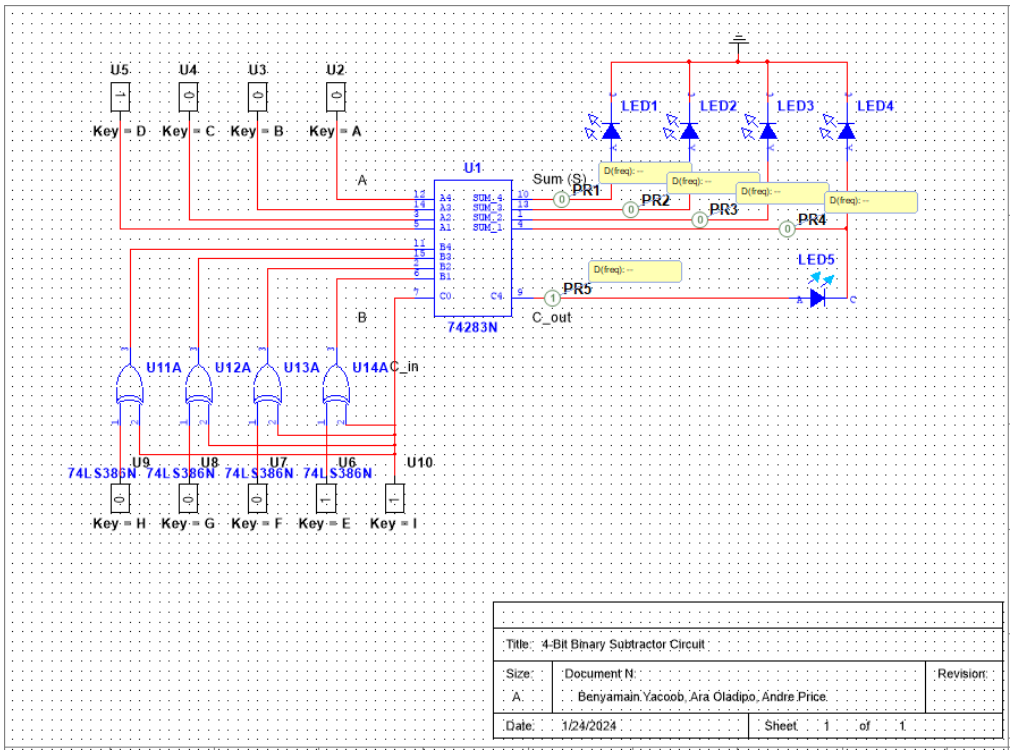

Figure 27: 4-Bit Binary Subtractor Demonstration 2

$$SUM_1 = A'B'C'D'EH' + A'B'C'FH' + A'CF'G'H + A'B'D'EFH' + A'E'F'G'H + A'CE'G'H + A'DF'G'H + A'CDG'H + A'B'GH' + A'C'D'EGH' + A'C'FGH' + A'BCF'H + A'D'EFGH' + A'BE'F'H + A'BCE'H + A'BDF'H + A'BG'H + A'BCDH + AB'C'D'EH + AB'C'FH + ACF'G'H' + AB'D'EFH + AE'F'G'H' + ACE'G'H' + ADF'G'H' + ACDG'H' + AB'GH + AC'D'EGH + AC'FGH + ABCF'H' + AD'EFGH + ABE'F'H' + ABCE'H' + ABDF'H' + ABG'H' + ABCDH' \quad (10)$$

$$SUM_2 = B'C'D'EG' + B'C'FG' + B'CF'G + B'D'EFG' + B'E'F'G + B'CE'G + B'DF'G + B'CDG + BC'D'EG + BC'FG + BCF'G' + BD'EFG + BE'F'G' + BCE'G' + BDF'G' + BCDG' \quad (11)$$

$$SUM_3 = C'D'EF' + C'E'F + C'DF + CD'EF + CE'F' + CDF' \quad (12)$$

$$SUM_4 = D'E + DE' \tag{13}$$

$$C_{out} = CF'G'H' + E'F'G'H' + CE'G'H' + DF'G'H' + CDG'H' + BCF'H' + BE'F'H' + BCE'H' + BDF'H' + BG'H' + BCDH' + ACF'G' + AE'F'G' + ACE'G' + ADF'G' + ACDG' + ABCF' + ABE'F' + ABCE' + ABDF' + ABG' + AH' + ABCD \tag{14}$$

7.2 Results and Analysis Discussion

The truth table generated by the logic converter is shown below:

Truth Table for 4-bit Subtractor: HEX outputs.

$M = 1$												
INPUTS								CALC OUT		ACTUAL OUT		Check if Correct
A_4	A_3	A_2	A_1	B_4	B_3	B_2	B_1	C_{out}	S_{out}	C_{out}	S_{out}	✓
0	1	0	1	0	1	0	1	0	0	1	0	
0	1	1	1	0	1	0	1	0	2	1	2	
1	0	0	1	0	1	0	1	0	4	1	4	
1	0	1	1	0	1	0	1	0	6	1	6	

It shows that we were right when it came to the SUM output, but we forgot about the CARRY output as we were calculating the truth table. Understanding how the CARRY output worked made us understand more the function of the binary subtractor. It also cemented the idea of the two's complement number. Images of our functioning circuits are shown below:

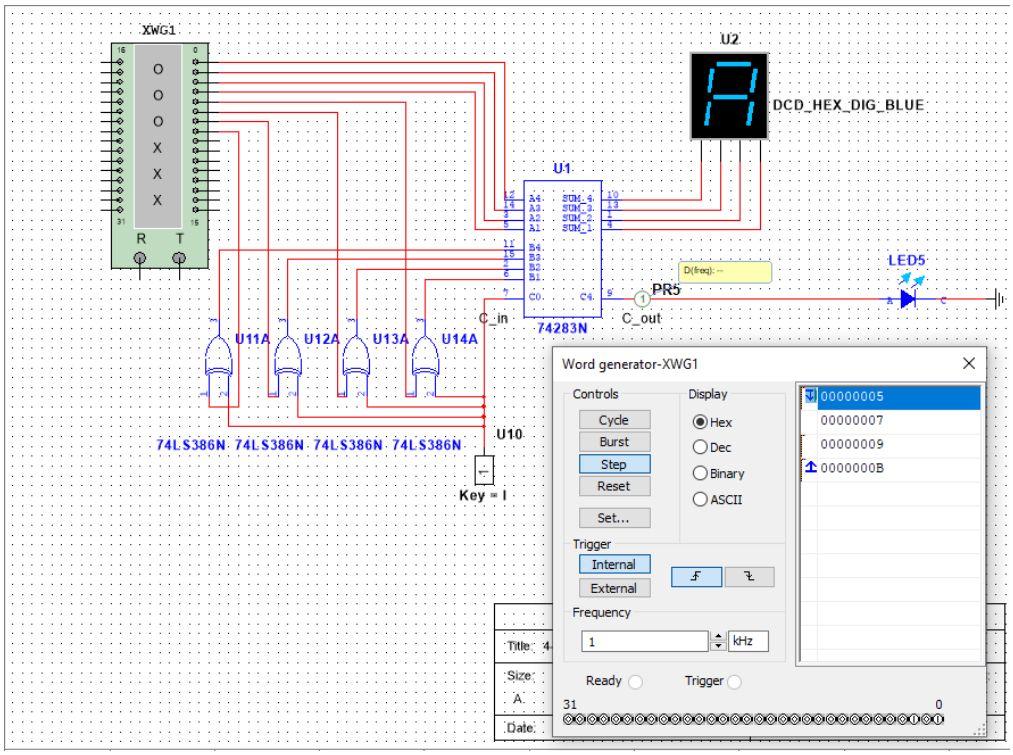

Figure 28: 4-Bit Binary Subtractor Word Generator A-Input Sequence 1

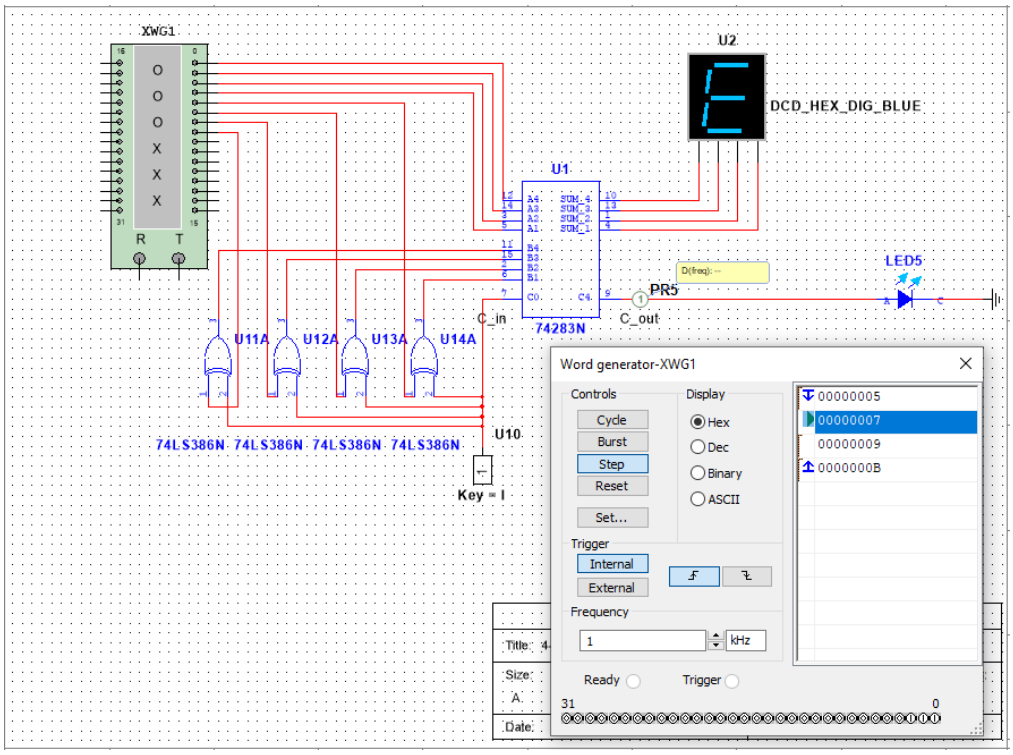

Figure 29: 4-Bit Binary Subtractor Word Generator A-Input Sequence 2

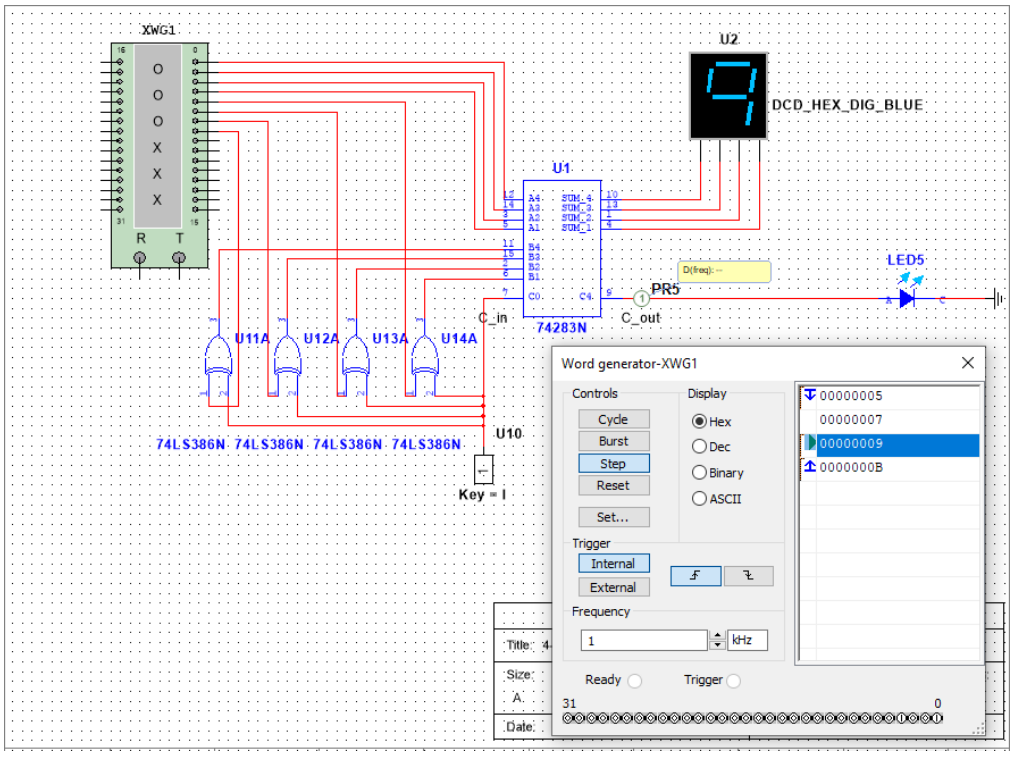

Figure 30: 4-Bit Binary Subtractor Word Generator A-Input Sequence 3

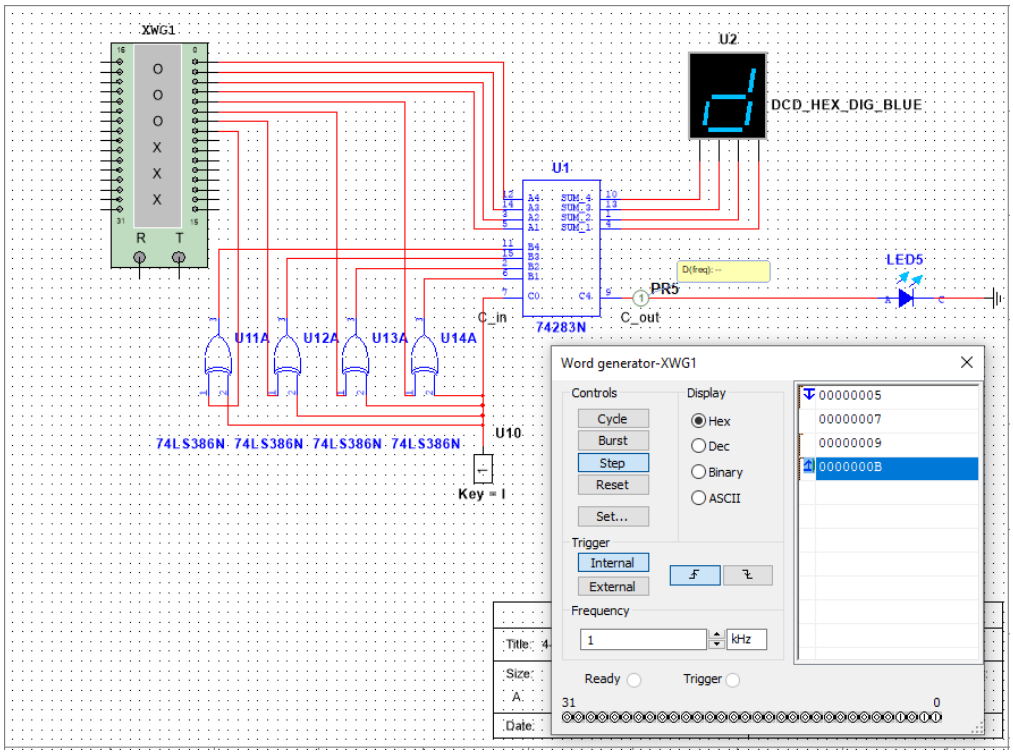

Figure 31: 4-Bit Binary Subtractor Word Generator A-Input Sequence 4

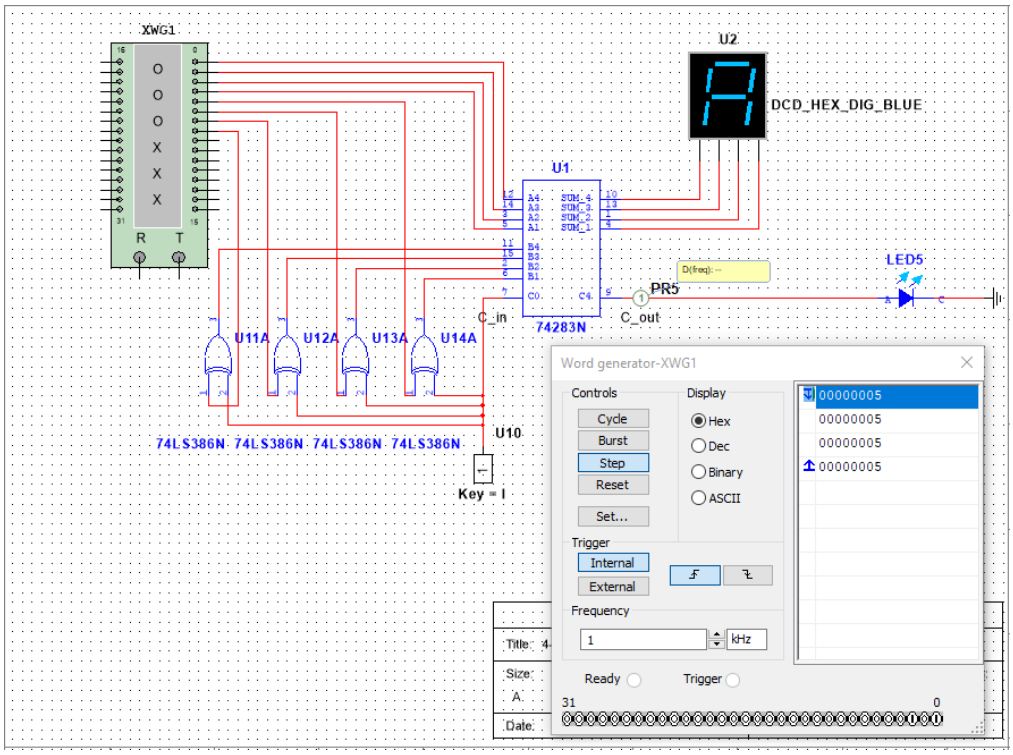

Figure 32: 4-Bit Binary Subtractor Word Generator B-Input Sequence

8 Results and Conclusions

This section discusses the results of the digital logic gate simulation experiments. The discussion provides in-depth explanations of the results and any discrepancies between the results and theoretical expectations. It also explores potential sources of error and discusses the accuracy of the tests, the debugging process, and what was learned from these experiences.

One of the biggest problems our group faced with this topic involved the use of the word generator. The word generator was a tool that took a large amount of effort, perseverance, and dedication to learn, and even now we still have to make the effort to continuously learn how to effectively use it. Due to our lack of familiarity with the tool, we attempted to learn how to use it by eyeing the diagram listed in the PDF for the second simulation assignment; consequently, this method did not work and only led to confusion. Thus, we attempted to look up multiple tutorials on the use of this tool, which was met with mixed results. Some tutorials were very effective: they gave what could be viewed as a guide on how to use the word generator. Other tutorials, however, were so advanced and beyond our realm of study or ability that they seemed to be useless as they pertained to our learning of the topic. Through

trial and error, we were able to learn the nuances of the tool; but doing so required a tremendous amount of effort to ascertain.

On a general scale, the time to complete this simulation report was less than that of the first assignment; however, it was juxtaposed by more complicated concepts which took up a large portion of the timescale used throughout the assignment. Having to learn the theory surrounding the complexities of adders and subtractors was something we managed to finish, thus the finalizing of this report, but not without working as a team to completely understand the topic.