ELEC 240
Lab 7 - Digital Logic

Test performed: October 23, 2018

Report submitted: November 2, 2018

Andrew Pham, Aneel Damaraju

1 Objective

The objective of this lab was to learn the underlying logical operations behind an adder and to then apply these principles to implement a one-bit full adder on a breadboard.

2 Materials

- Test Board
- 1x 74HC00 Quad 2-input NAND gate
- 1x 7486 Quad 2-input XOR gate
- 2x LEDs
- 1x Quad DIP switch
- 5x 330Ω resistors
- Yellow, Red, and Black wire (signal, power, gnd)
- Wire strippers and cutters

3 Test Description

In the first section of the lab, we learned about the logic underlying the fundamental NAND and XOR gates and their truth tables for all combinations of inputs. We then formulated an expression for the correct output of the adder for the three input bits. We then applied DeMorgan's law to simplify the circuit so that it only uses NAND and XOR gates, and we then constructed the one-bit full adder.

3.1 Pre-Lab Calculations and Schematics

To begin, we had to understand the underlying logic behind a NAND and a XOR gate, which the adder was composed of. Below in Figure 1 is the truth table for a NAND gate, and in Figure 2 is the truth table for a XOR gate.

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Figure 1: NAND Truth Table

A	B	A NAND B
0	0	0
0	1	1
1	0	1
1	1	0

Figure 2: XOR Truth Table

We then derived the following expression for the Sum S and Carry C output bits of the adder based on the three one-bit inputs x, y, z.

$$S = (x \bigoplus y) \bigoplus z$$

$$C = (x \bullet y) \vee (x \bigoplus y) \bullet z$$

We then created the following truth table for the sum and carry bits based on these outputs as follows in Fig 3:

x	y	z	$x \bullet y$	$x \bigoplus y$	$(x \bigoplus y) \bigoplus z$	$(x \bullet y) \vee (x \bigoplus y)$	C	S
0	0	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	1
0	1	0	0	1	1	1	0	1
0	1	1	0	1	0	1	1	0
1	0	0	0	1	1	1	0	1
1	0	1	0	1	0	1	1	0
1	1	0	0	0	1	0	0	1
1	1	1	1	0	1	1	1	1

Figure 3: Naive Truth Table for the Sum and Carry bits

However, the direct implementation of the above truth table into logic requires the use of 2 XOR gates for the sum bit 2 NAND gates, and 1 OR gate for the carry bit. Since we want to reduce the complexity of our adder, we can use DeMorgan's laws to simplify the requirements for the carry bits:

$$x \bullet y = \neg(x \vee y)$$

$$x \vee y = \neg(x \bullet y)$$

We then obtain the following equation for the carry bit expression by applying the above DeMorgan's laws:

$$C = \neg \left(\neg(x \bullet y) \bullet \neg \left(\left(x \bigoplus y \right) \bullet z \right) \right)$$

The truth table is depicted below in Fig 4;

x	y	z	$x \bullet y$	$x \bigoplus y$	$\neg((x \bigoplus y) \bullet z)$	$\neg(x \bullet y) \bullet \neg((x \bigoplus y) \bullet z)$	C
0	0	0	0	0	1	1	0
0	0	1	0	0	1	1	0
0	1	0	0	1	1	1	0
0	1	1	0	1	0	0	1
1	0	0	0	1	1	1	0
1	0	1	0	1	0	0	1
1	1	0	0	0	1	1	0
1	1	1	1	0	1	0	1

Figure 4: Modified Truth Table for the Sum and Carry bits

The circuit that implements this truth table's equation is depicted below in Fig 5 :

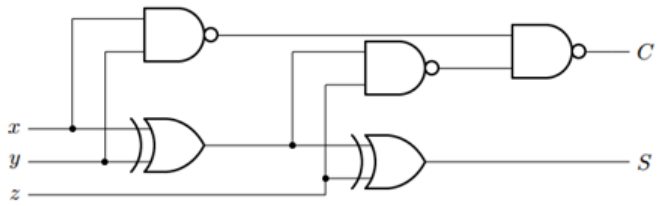

Figure 5: One-Bit Full Adder Circuit Diagram

4 Results and Discussion

4.1 Part A: Understanding the components

Before understanding an adder, it is important to learn what the components are and what they do. The list of components used in the circuit are mentioned in the materials, with the relevant logic materials being the LED, Quad DIP switch and XOR and NAND gate. The LED is useful as a display, as having two LED's effectively displays all possible two bit combinations from 00 to 11 in binary or 0 to 3 in decimal. The Quad DIP switch is used to convey which bits will be added, up to a maximum of three. The XOR and NAND gates perform their desired intended logical function on the two input bits, and will be implemented as parts of the adder.

4.2 Part B: The Full-Adder Circuit

This section was pretty straight forward, but was an attempt to understand how to implement a full adder in terms of the circuit components. The truth table for the full-adder can be seen in Figure 3. However, this table does not take into account the current physical limitations, of only having NAND and XOR gates. This can be seen in Figure 4 which has taken into account DeMorgan's law and thus only uses NAND and XOR gates.

4.3 Part C: Constructing the Full-Adder

As the name implies, this section is devoted to using by creating the Full-Adder circuit physically. The steps taken toward creating the logical circuit can be seen in Figure 5. The only steps needed after creating this are attaching the x,y and z bits as well as displaying the carry and sum bit outputs. The inputs are created by attaching 3 of the 4 switches on the quad DIP to power and ground, with each of the switches acting as a

different input bit. For the output bits, connecting an led resistor combination through each of the outputs will create a 1 or a 0 bit. The output for each of the bits can be seen below.

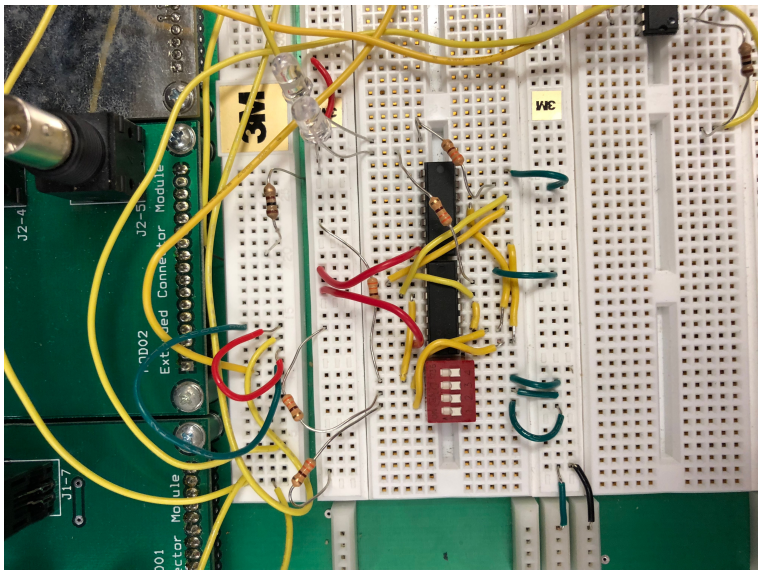

Figure 6: 000 Input with 00 Output

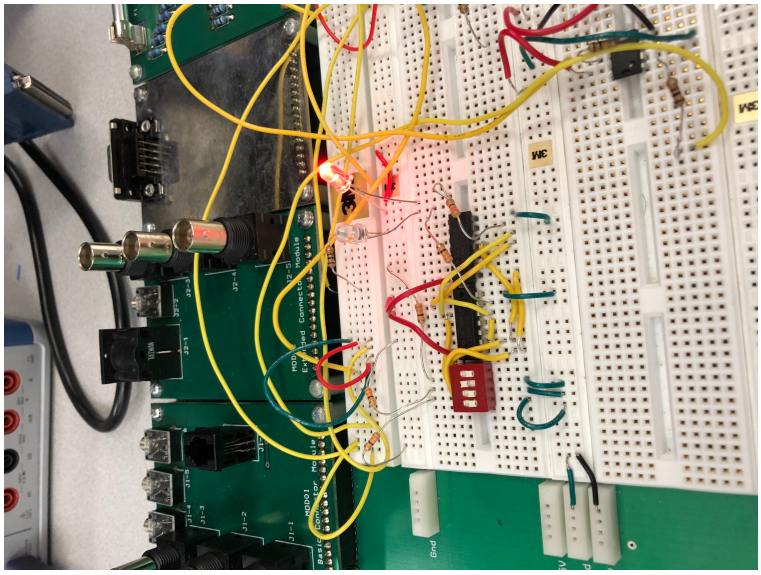

Figure 7: 001 Input with 01 Output

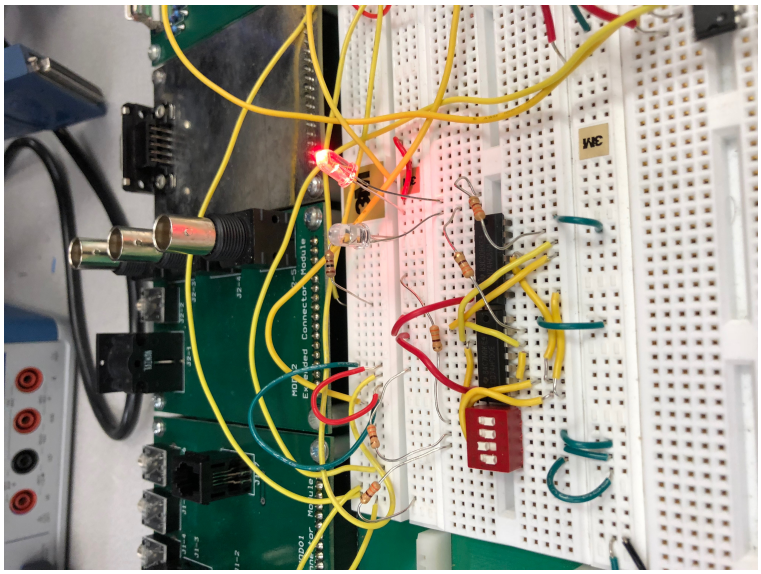

Figure 8: 010 Input with 01 Output

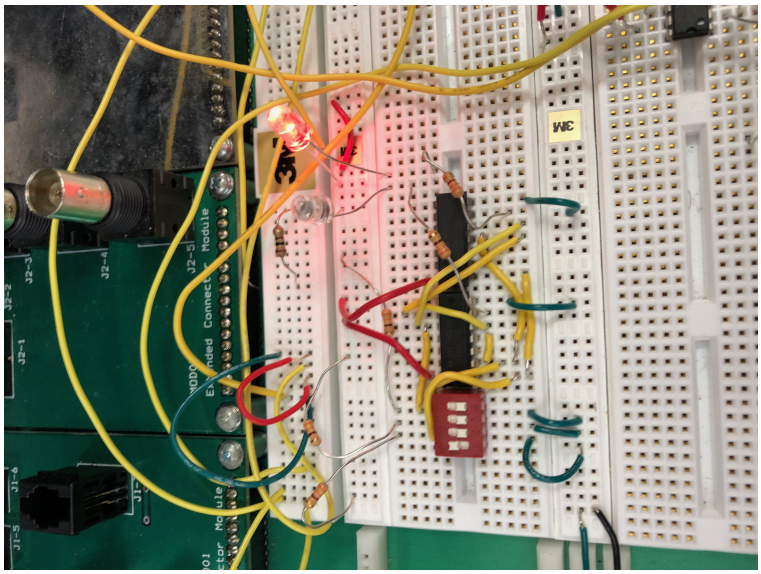

Figure 9: 100 Input with 01 Output

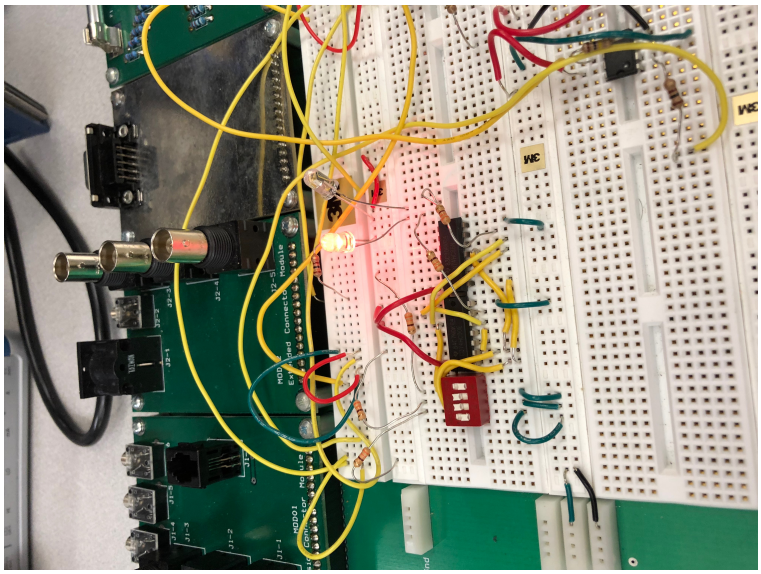

Figure 10: 011Input with 10 Output

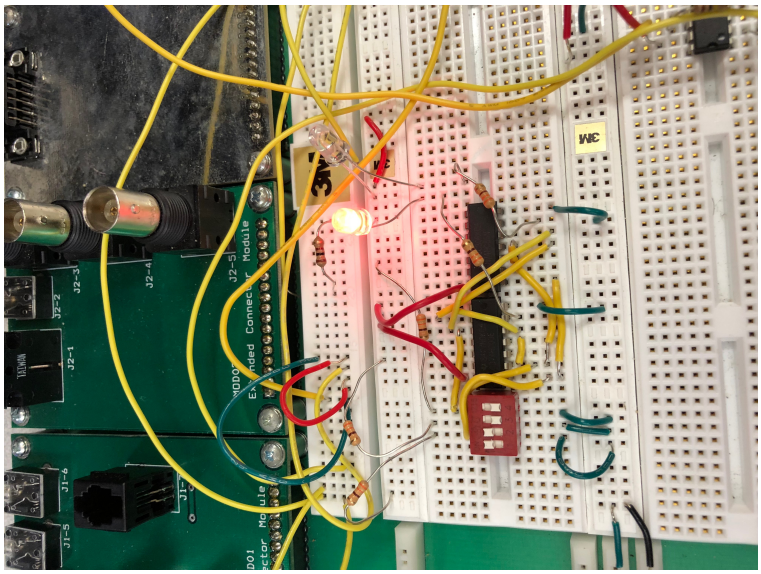

Figure 11: 110 Input with 10 Output

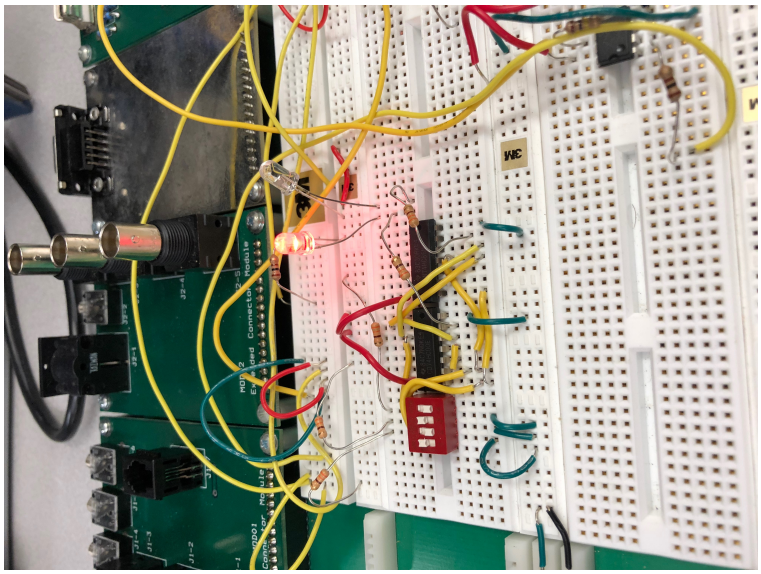

Figure 12: 101 Input with 10 Output

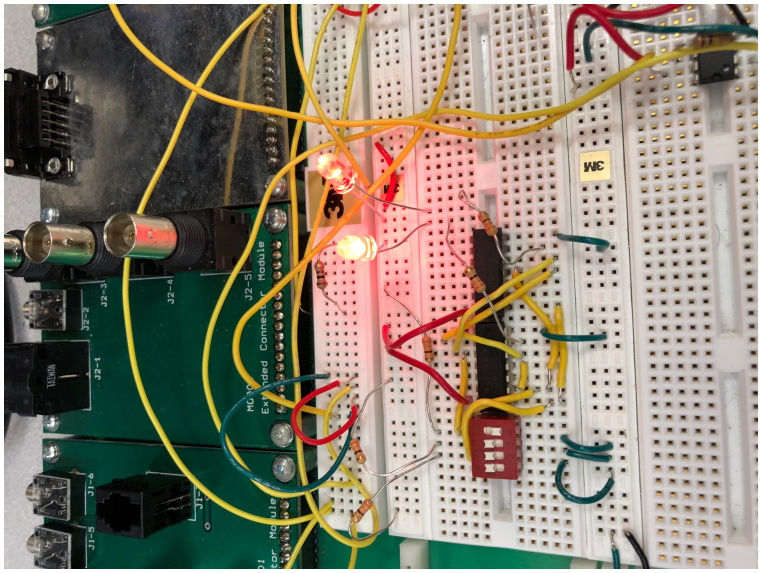

Figure 13: 111 Input with 11 Output

5 References

https://www.ece.rice.edu/ dpr2/elec240/lab7/experiment_7-1/

6 Conclusion

In part A, we learned the various circuit elements and truth tables needed to create a full adder. By adding various truth tables in series, a three bit adder can be created with a carry and sum term. In part B, we combined the truth tables into two that had the desired carry and sum terms. However, this was not done

in terms of NAND and XOR, so using DeMorgan's law, these tables were modified to work with only NAND and XOR components. In Part C, we physically created the circuit, and the successful outputs were shown in the Results and Discussion section. Using the given components, the logic checked out, and the full-adder was successful.

7 Errors

Since this was a relatively straightforward lab with boolean outputs that returned what was planned, so no errors were experienced. However, if the outputs do not come out as planned possible errors can be attributed to poor wire management or human error.