Syed Ali Zaidi
zaidis33

Lab 6

Contents

Deliverables 1
Introduction 2
Logic Expression 2
Analytical Solution 2
  Sum of Products Implementation 3
  Product of Sums Implementation 5
Digital Solution 7
  AND/OR/INVERTER 7
  NAND only 8
Physical Solution 16
Discussion 22
Reflection 24

Deliverables

1. Choose an example desired logic function that takes 4 binary inputs to implement like in the K-mapping videos (i.e., 4 input bits and 1 output bit)
2. Show the k-mapping process you used to come up with a logic expression you can implement with gates.
3. Use Boolean algebra to then implement this two different ways:
   a. Using a combination of AND gates, OR gates, and INVERTERs, and
   b. Using *only* NAND gates.
4. For each of the two gate implementations from steps 3a and 3b:
   a. Analytically check that it works by generating a truth table from the gates and comparing with the desired behaviour for your logic function,
   b. Build the circuit in multisim and demonstrate that it produces the desired functionality, and
   c. Physically implement the circuit and show that it produces the desired functionality.

Syed Ali Zaidi
zaidis33

Introduction

For this lab, I will present my solution three ways. I will first solve the circuit analytically in which I shall use the logic expression to make a truth table. The truth table will be used to make a K-map for both the sum of products and product of sum implementation. Then I shall replicate the circuit in Multisim which will have two versions: one with AND/OR/INVERTER gates and the other with only NAND gates. For my physical solution, I will make the circuit on two breadboards, and, like the digital solution, I will have two versions of it.

Logic Expression

Given a 4-bit binary number which is guaranteed to be in the range [1,14], output a 1 if and only if the number is a factor of 6.

Analytical Solution

With the analytical solution, I will first write the numbers which give a high output and low output. The high output numbers represent the numbers which are TRUE (or have a value of 1) with respect to the logic expression. The low output numbers are numbers which return FALSE (or have a value of 0) with respect to the logic expression. The numbers we shall ignore are those which are not in the range provided in the logic expression.

High Output Numbers = 1, 2, 3, 6

Low Output Numbers = 4, 5, 7, 8, 9, 10, 11, 12, 13, 14

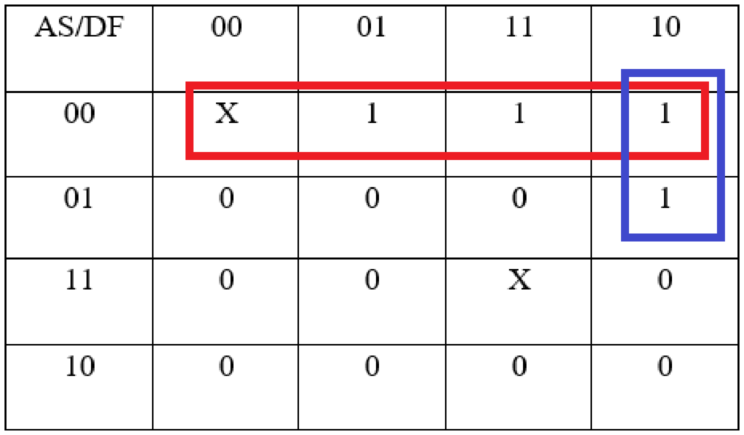
Ignore = 0, 15

I then summarized the results into a truth table. ASDF represent the four-digit binary numbers. For example, in 0110 the value of A is 0, S is 1, D is 1 and F is 0. Note, the Xs represent the numbers we have ignored.

AS/DF	00	01	11	10
00	X	1	1	1
01	0	0	0	1
11	0	0	X	0
10	0	0	0	0

Sum of Products Implementation

To determine an expression for logic expression we shall use the sum of products method. The first step of this is to make a K-map. This is shown below. I have drawn the largest possible rectangle (covering a space of 2^n) around the high output numbers.



AS/DF	00	01	11	10
00	X	1	1	1
01	0	0	0	1
11	0	0	X	0
10	0	0	0	0

Now that all the high output numbers are enclosed in the rectangles, we can derive an expression for the red and blue box.

$$\text{Red Box} = \bar{A}\,\bar{S}$$

$$\text{Blue Box} = \bar{A}D\bar{F}$$

Summing these products (taking the OR) will give us:

Syed Ali Zaidi
zaidis33

$$\bar{A}\,\bar{S} + \bar{A}D\bar{F}$$

We shall now make a table which will highlight exactly how the output will change across every possible input.

I will demonstrate the Boolean Algebra I used for calculating the output of 0110 (decimal 6).

$$\bar{A}\,\bar{S} + \bar{A}D\bar{F}$$

$$\overline{(0)} \cdot \overline{(1)} + \overline{(0)} \cdot (1) \cdot \overline{(0)}$$

$$1 \cdot 0 \;+\; 1 \cdot 1 \cdot 1$$

$$0 \;+\; 1 \;=\; 1$$

Thus, an output of TRUE will be returned.

Using the same method as above I repeated it for all the other inputs.

Decimal Number	A	S	D	F	Output
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

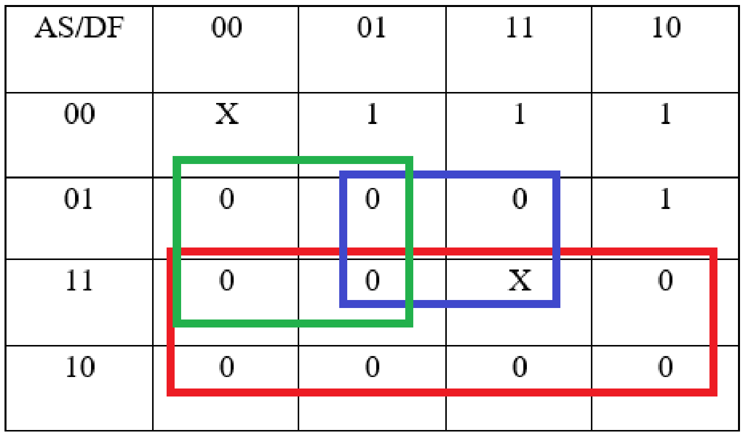
As seen above, the analytical expressions results give the same outputs as the truth table meaning that the expression is correct. Also note that the above table includes 0 and 15 even though they were ignored in the truth table. This is just for completion and will not be considered.

Syed Ali Zaidi
zaidis33

Product of Sums Implementation

Like the sum of products, we can also use the product of sums to find an expression. The difference over here is that we will use the low input numbers instead of the high input ones; the algorithm will remain the same.

Below is the K-map for the product of sums implementation:



AS/DF	00	01	11	10
00	X	1	1	1
01	0	0	0	1
11	0	0	X	0
10	0	0	0	0

As seen above, the green, blue, and red rectangles have enclosed 2^n numbers and cover the largest amount of area possible.

The expression can be seen as:

Red Box = $\bar{A}$

Blue Box = $\bar{S} + \bar{F}$

Green Box = $\bar{S} + D$

$$(\bar{A})\,(\bar{S} + \bar{F})\,(\bar{S} + D)$$

Syed Ali Zaidi
zaidis33

We shall now make a table of how the output changes with regards to a range of inputs. I will demonstrate 0101 (decimal 5) as an example:

$$(\bar{A})\ (\bar{S}\ +\ \bar{F})\ (\bar{S}\ +\ D)$$

$$(\bar{0})\ (\bar{1}\ +\ \bar{1})\ (\bar{1}\ +\ 0)$$

$$(1)(0+0)(0+0)$$

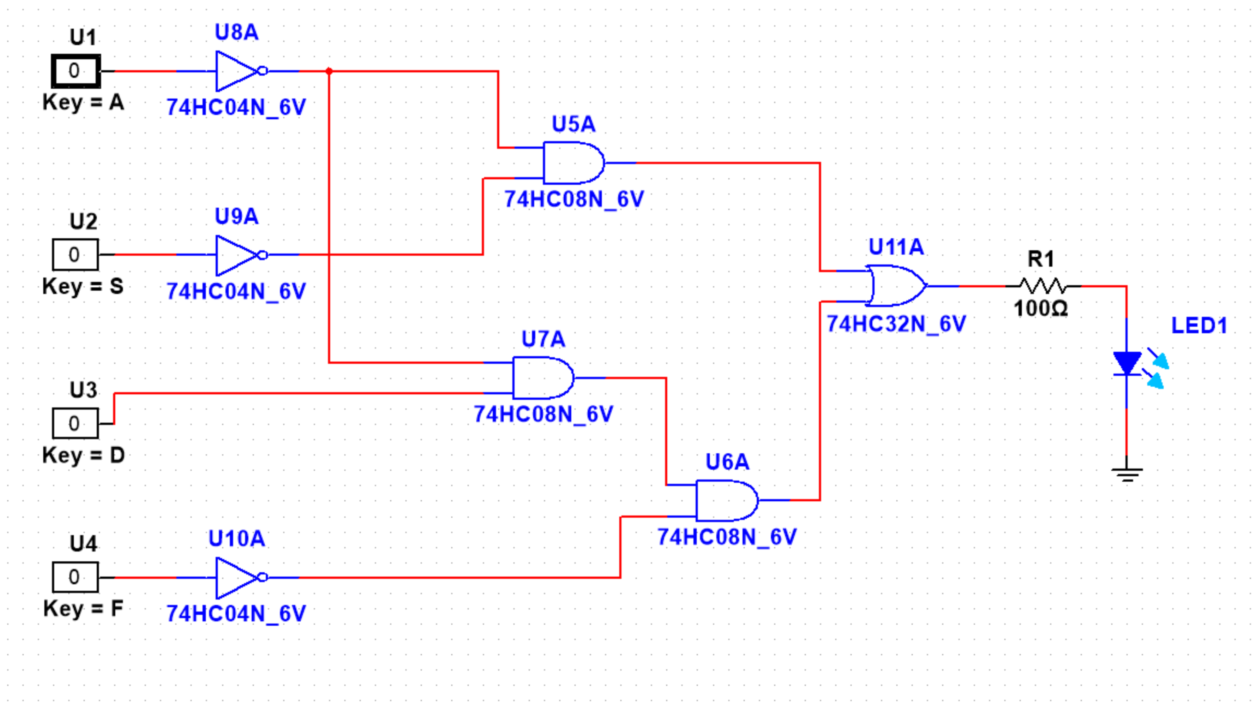
$$(1)(0)(0)=0$$

Decimal Number	A	S	D	F	Output
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

Digital Solution

For my digital solution I will use the expression derived in the sum of products to replicate it in Multisim. I will present my digital solution two ways. The first one will involve implementation using the AND/OR/INVERTER gates. The second one will only use NAND gates. Note, I have only shown the NAND conversion for the sum of products as I will be using only the sum of products expression (according to the flexibility given to us in the deliverables).

AND/OR/INVERTER

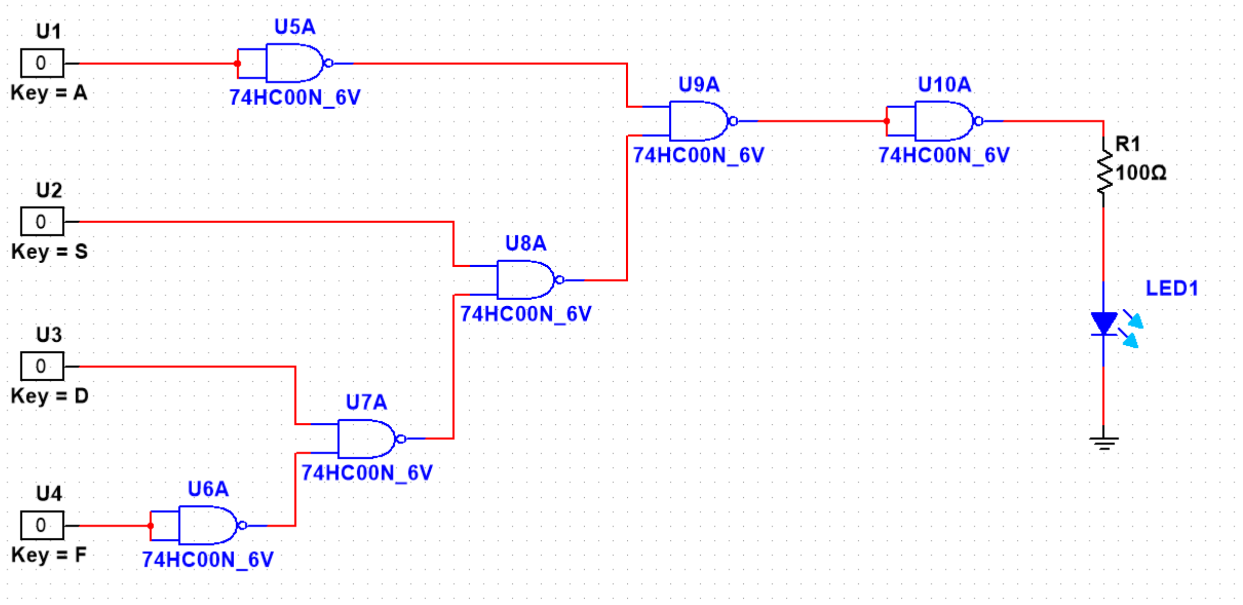


The formula we are using is $\bar{A}\,\bar{S} + \bar{A}D\bar{F}$. This means that we need one OR gate, three AND gates and three NOT gates. The table below shows the outputs based on my theoretical knowledge of gates which shows that the gates chosen will give the correct output theoretically.

Syed Ali Zaidi
zaidis33

Truth Table (Theoretical)										
A	S	D	F	U8A	U9A	U5A	U7A	U10A	U6A	U11A (Output)
0	0	0	0	1	1	1	0	1	0	1
0	0	0	1	1	1	1	0	0	0	1
0	0	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	1	0	0	0	1	0	0
0	1	0	1	1	0	0	0	0	0	0
0	1	1	0	1	0	0	1	1	1	1
0	1	1	1	1	0	0	1	0	0	0
1	0	0	0	0	1	0	0	1	0	0
1	0	0	1	0	1	0	0	0	0	0
1	0	1	0	0	1	0	0	1	0	0
1	0	1	1	0	1	0	0	0	0	0
1	1	0	0	0	0	0	0	1	0	0
1	1	0	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0	0	0	0

NAND only



Syed Ali Zaidi
zaidis33

Now we shall work out the digital solution using only NAND gates. We shall do this by performing Boolean Algebra manipulations on our expression.

$$\bar{A}\,\bar{S} + \bar{A}D\bar{F}$$

$$\bar{A}(\bar{S} + D\bar{F})$$

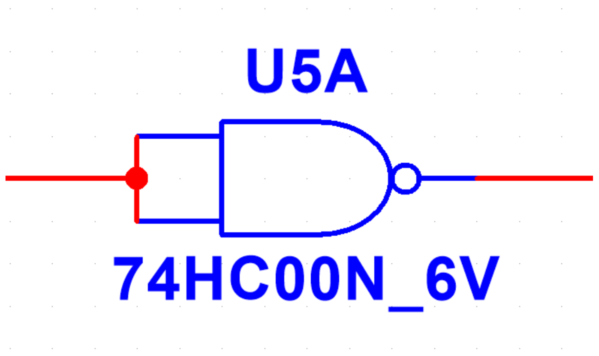
$$\bar{A}\overline{\overline{(\bar{S} + D\bar{F})}}$$

$$\bar{A}\,(\overline{\bar{\bar{S}}\,\overline{D\bar{F}}})$$

$$\bar{A}\,(\overline{S\,\overline{D\bar{F}}})$$

$$\overline{\overline{\bar{A}\,(\overline{S\,\overline{D\bar{F}}})}}$$

The expression we determined shows us that we need 6 NAND gates in total. Note that the NOT gate is replicated by connecting the two inputs together as shown below:



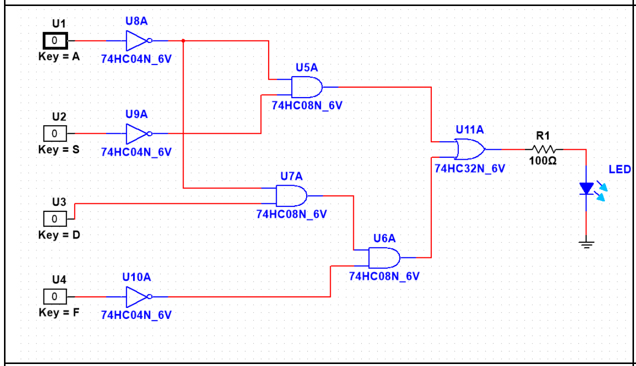
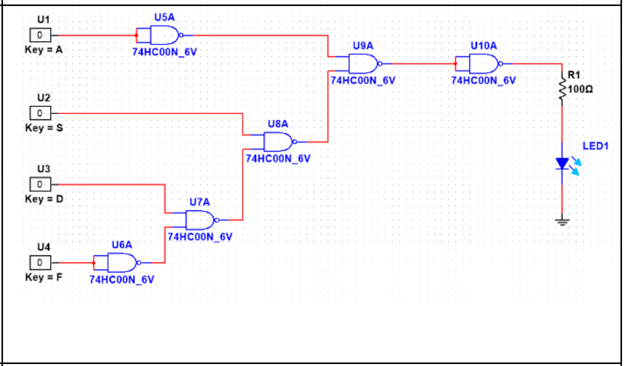
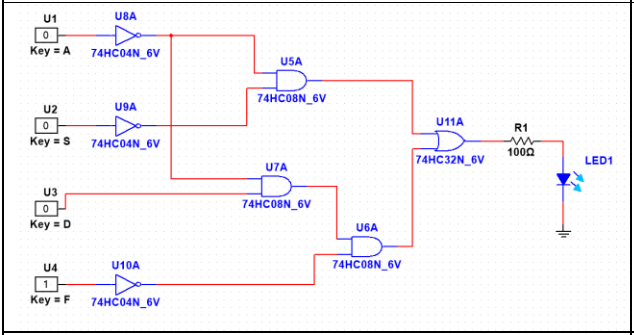
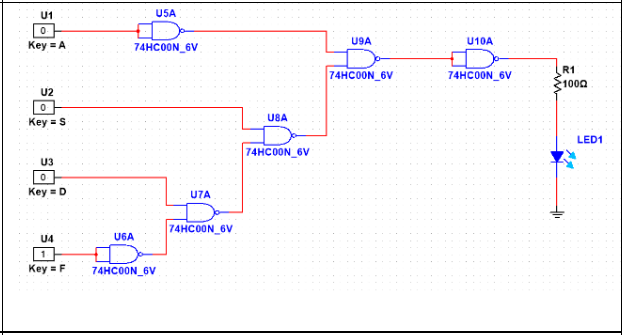
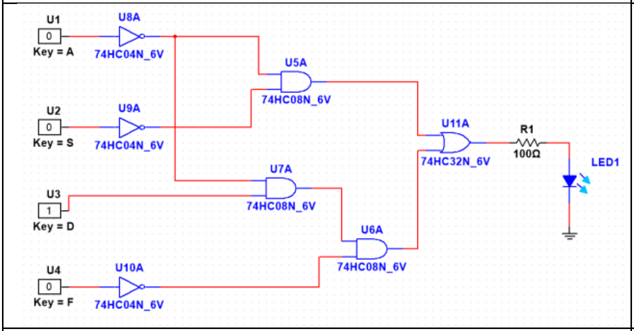
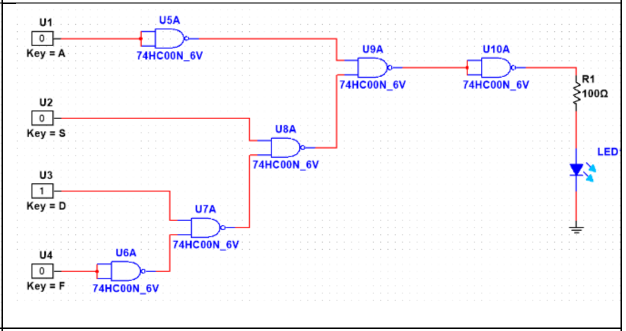
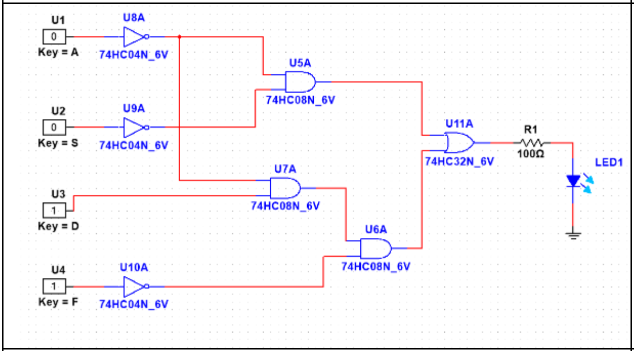
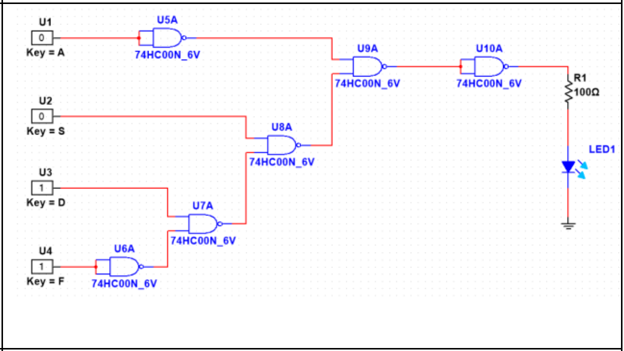
The truth table below shows the theoretical determination of what the outputs will be according to our NAND gate knowledge.

Syed Ali Zaidi
zaidis33

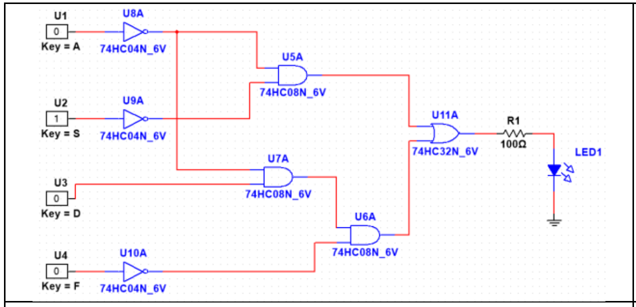
A	S	D	F	U6A	U7A	U8A	U5A	U9A	U10A (Output)
0	0	0	0	1	1	1	1	0	1
0	0	0	1	0	1	1	1	0	1
0	0	1	0	1	0	1	1	0	1
0	0	1	1	0	1	1	1	0	1
0	1	0	0	1	1	0	1	1	0
0	1	0	1	0	1	0	1	1	0
0	1	1	0	1	0	1	1	0	1
0	1	1	1	0	1	0	1	1	0
1	0	0	0	1	1	1	0	1	0
1	0	0	1	0	1	1	0	1	0
1	0	1	0	1	0	1	0	1	0
1	0	1	1	0	1	1	0	1	0
1	1	0	0	1	1	0	0	1	0
1	1	0	1	0	1	0	0	1	0
1	1	1	0	1	0	1	0	1	0
1	1	1	1	0	1	0	0	1	0

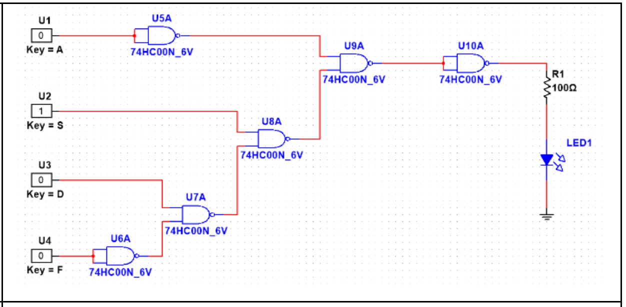
Below is a table showing all the possible inputs and their outputs (in the form of the switching on of an LED).

Syed Ali Zaidi
zaidis33

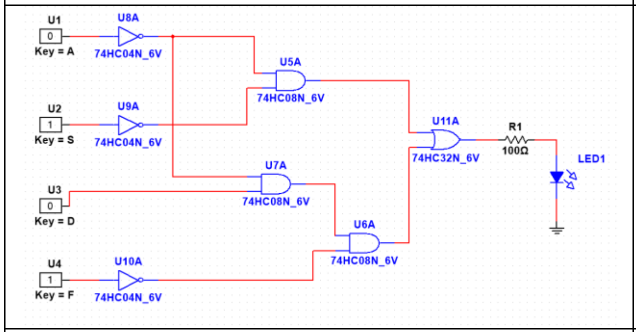
AND/OR/INVERTER	NAND
0000	0000
	
0001	0001
	
0010	0010
	
0011	0011
	
0100	0100

Syed Ali Zaidi
zaidis33

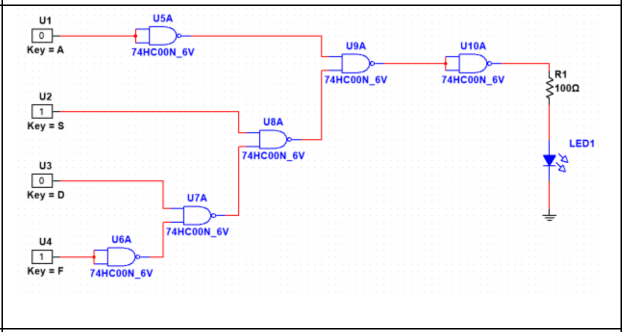




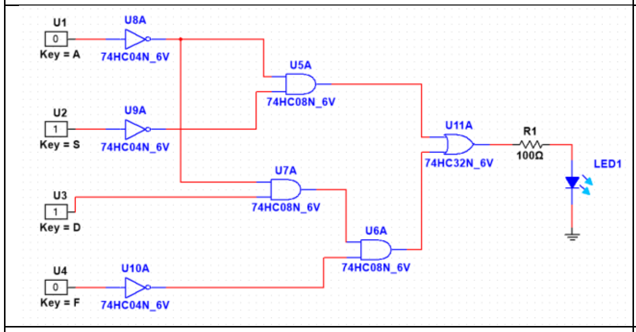
0101



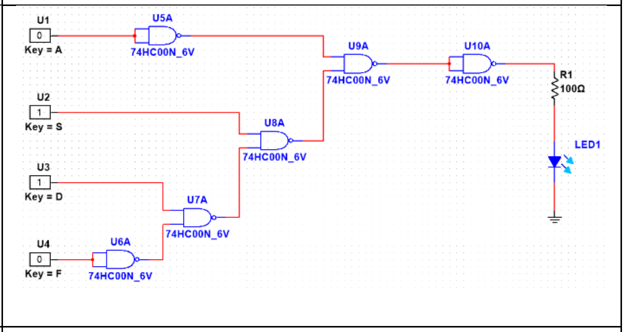
0101



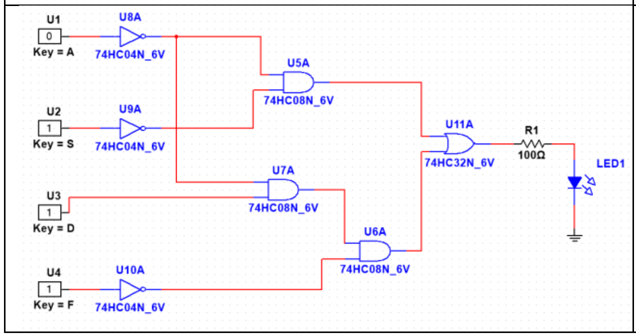
0110



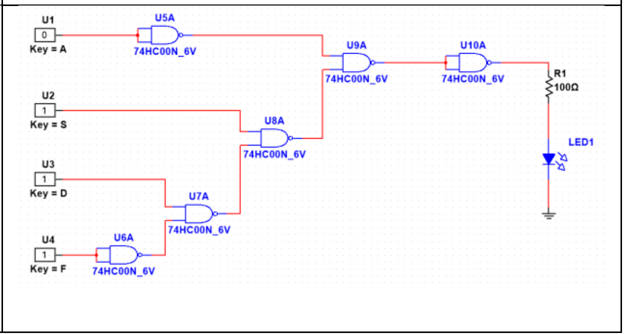
0110



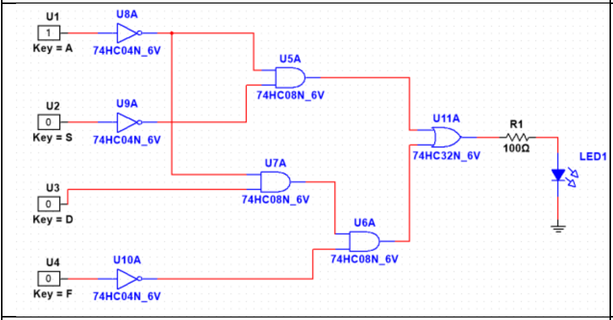
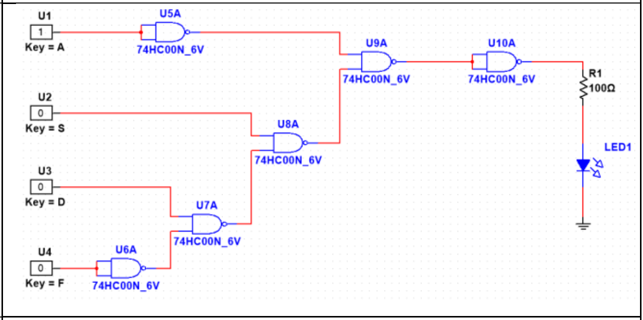
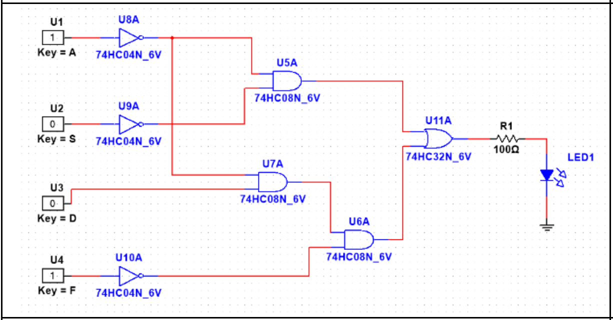
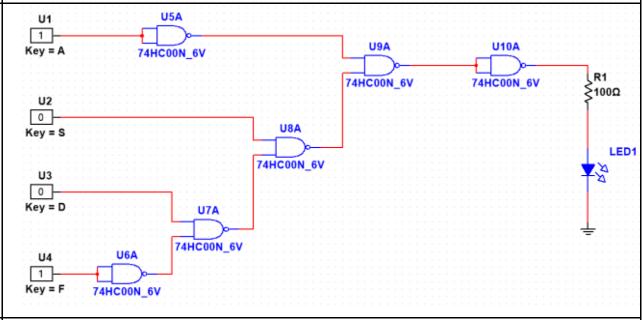
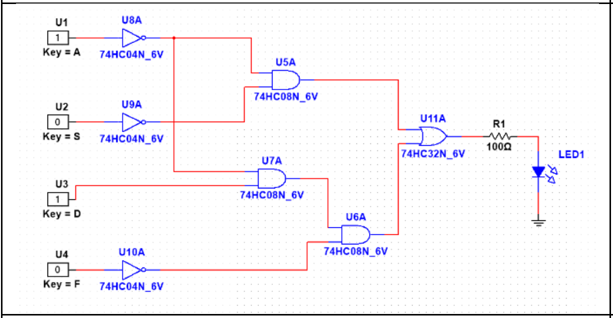
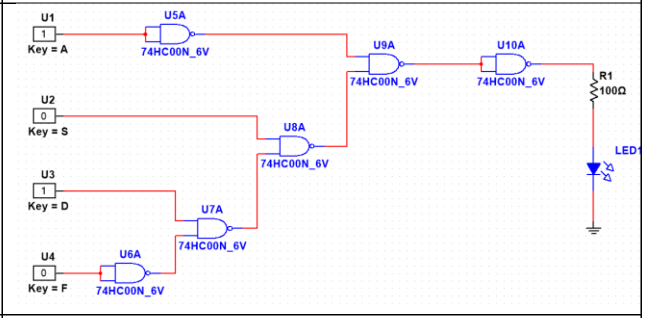
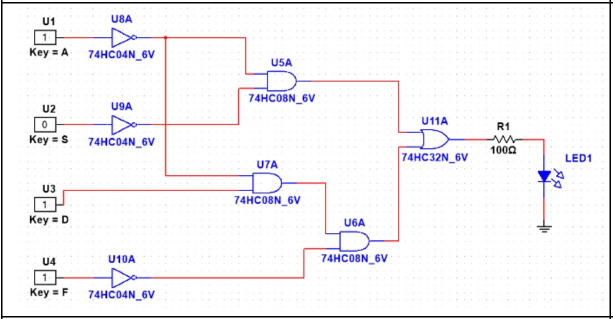
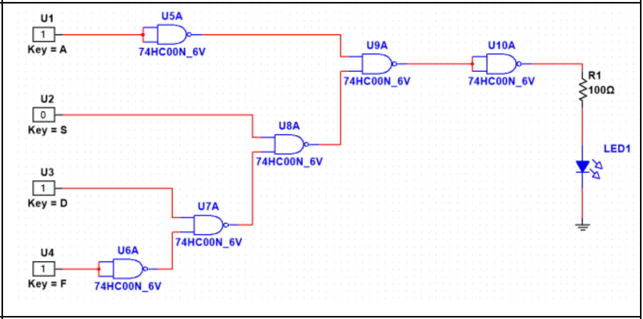
0111



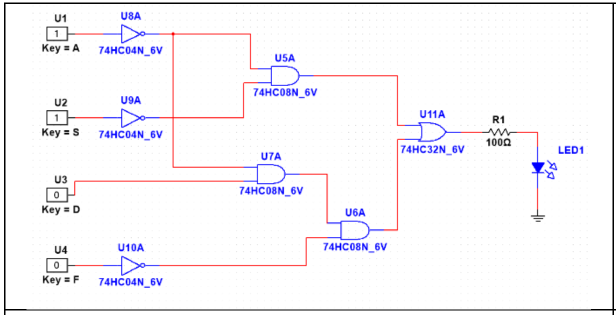
0111

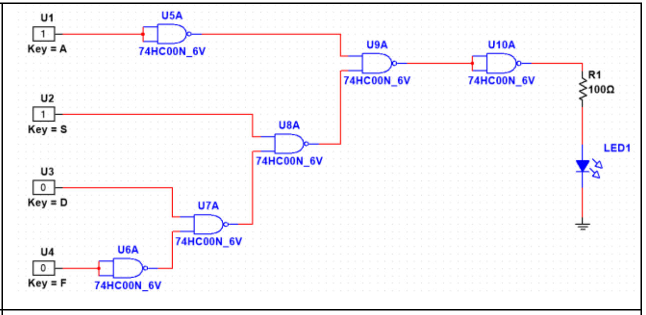


Syed Ali Zaidi
zaidis33

1000	1000
 U1 1 Key = A U8A 74HC04N_6V U2 0 Key = S U9A 74HC04N_6V U5A 74HC08N_6V U7A 74HC08N_6V U3 0 Key = D U6A 74HC08N_6V U4 0 Key = F U10A 74HC04N_6V U11A 74HC32N_6V R1 100Ω LED1	 U1 1 Key = A U5A 74HC00N_6V U9A 74HC00N_6V U10A 74HC00N_6V R1 100Ω LED1 U2 0 Key = S U8A 74HC00N_6V U3 0 Key = D U7A 74HC00N_6V U4 0 Key = F U6A 74HC00N_6V
1001	1001
 U1 1 Key = A U8A 74HC04N_6V U2 0 Key = S U9A 74HC04N_6V U5A 74HC08N_6V U7A 74HC08N_6V U3 0 Key = D U6A 74HC08N_6V U4 1 Key = F U10A 74HC04N_6V U11A 74HC32N_6V R1 100Ω LED1	 U1 1 Key = A U5A 74HC00N_6V U9A 74HC00N_6V U10A 74HC00N_6V R1 100Ω LED1 U2 0 Key = S U8A 74HC00N_6V U3 0 Key = D U7A 74HC00N_6V U4 1 Key = F U6A 74HC00N_6V
1010	1010
 U1 1 Key = A U8A 74HC04N_6V U2 0 Key = S U9A 74HC04N_6V U5A 74HC08N_6V U7A 74HC08N_6V U3 1 Key = D U6A 74HC08N_6V U4 0 Key = F U10A 74HC04N_6V U11A 74HC32N_6V R1 100Ω LED1	 U1 1 Key = A U5A 74HC00N_6V U9A 74HC00N_6V U10A 74HC00N_6V R1 100Ω LED1 U2 0 Key = S U8A 74HC00N_6V U3 1 Key = D U7A 74HC00N_6V U4 0 Key = F U6A 74HC00N_6V
1011	1011
 U1 1 Key = A U8A 74HC04N_6V U2 0 Key = S U9A 74HC04N_6V U5A 74HC08N_6V U7A 74HC08N_6V U3 1 Key = D U6A 74HC08N_6V U4 1 Key = F U10A 74HC04N_6V U11A 74HC32N_6V R1 100Ω LED1	 U1 1 Key = A U5A 74HC00N_6V U9A 74HC00N_6V U10A 74HC00N_6V R1 100Ω LED1 U2 0 Key = S U8A 74HC00N_6V U3 1 Key = D U7A 74HC00N_6V U4 1 Key = F U6A 74HC00N_6V
1100	1100

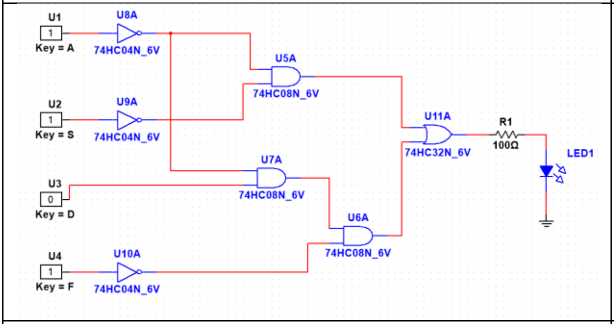
Syed Ali Zaidi
zaidis33

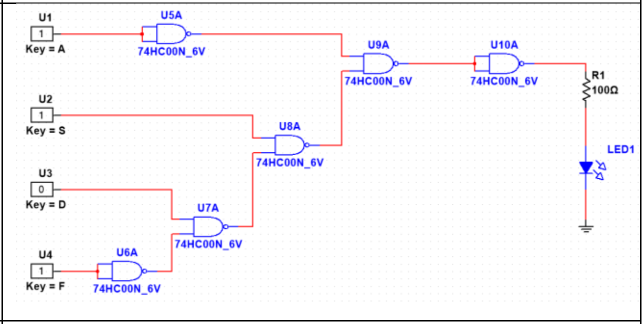




1101

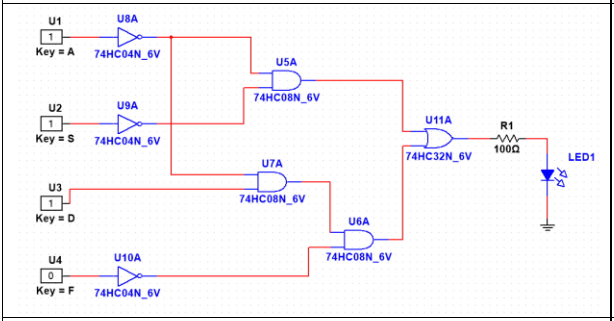
1101

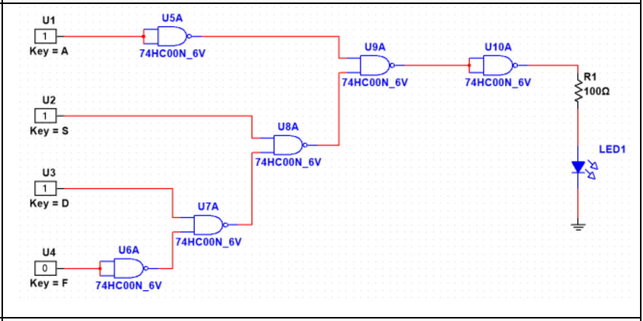




1110

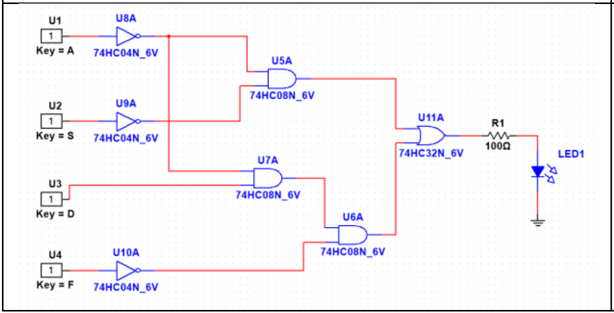
1110

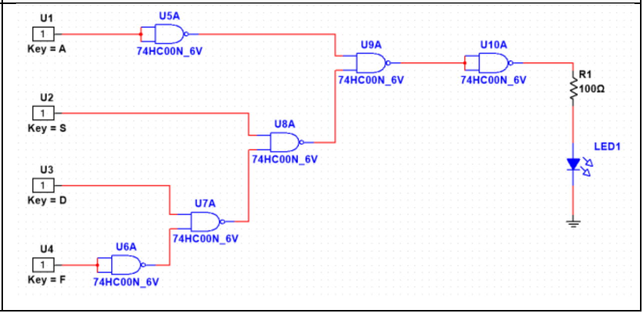




1111

1111





Syed Ali Zaidi
zaidis33

These outputs can be formulated into a table and compared with the analytical solution.

Decimal Number	A	S	D	F	Digital Output	Analytical Output
0	0	0	0	0	1	1
1	0	0	0	1	1	1
2	0	0	1	0	1	1
3	0	0	1	1	1	1
4	0	1	0	0	0	0
5	0	1	0	1	0	0
6	0	1	1	0	1	1
7	0	1	1	1	0	0
8	1	0	0	0	0	0
9	1	0	0	1	0	0
10	1	0	1	0	0	0
11	1	0	1	1	0	0
12	1	1	0	0	0	0
13	1	1	0	1	0	0
14	1	1	1	0	0	0
15	1	1	1	1	0	0

As seen above, the outputs from both the solutions are the same meaning that my digital solution results are valid.

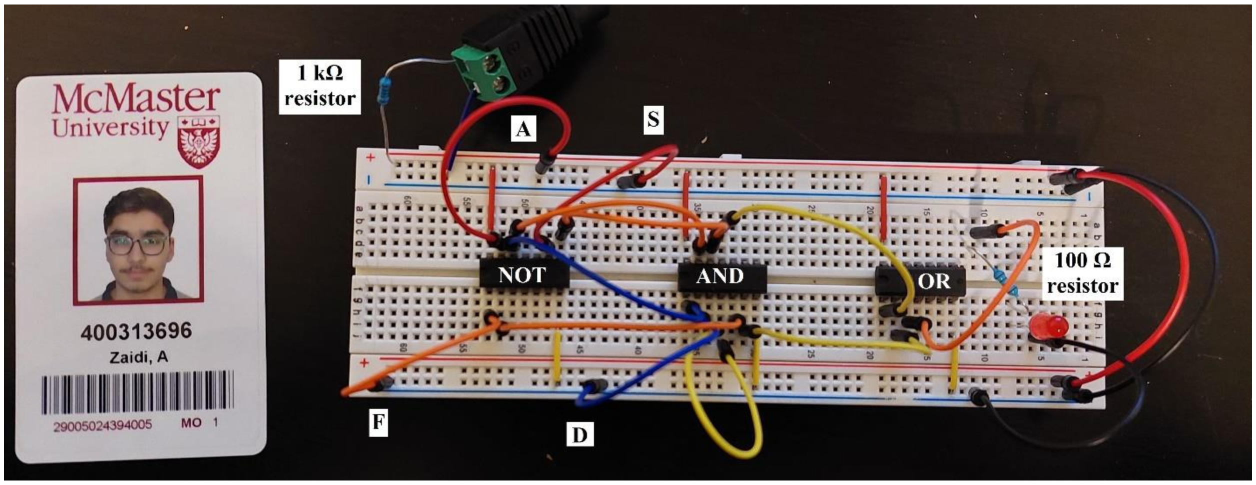
Syed Ali Zaidi
zaidis33

Physical Solution

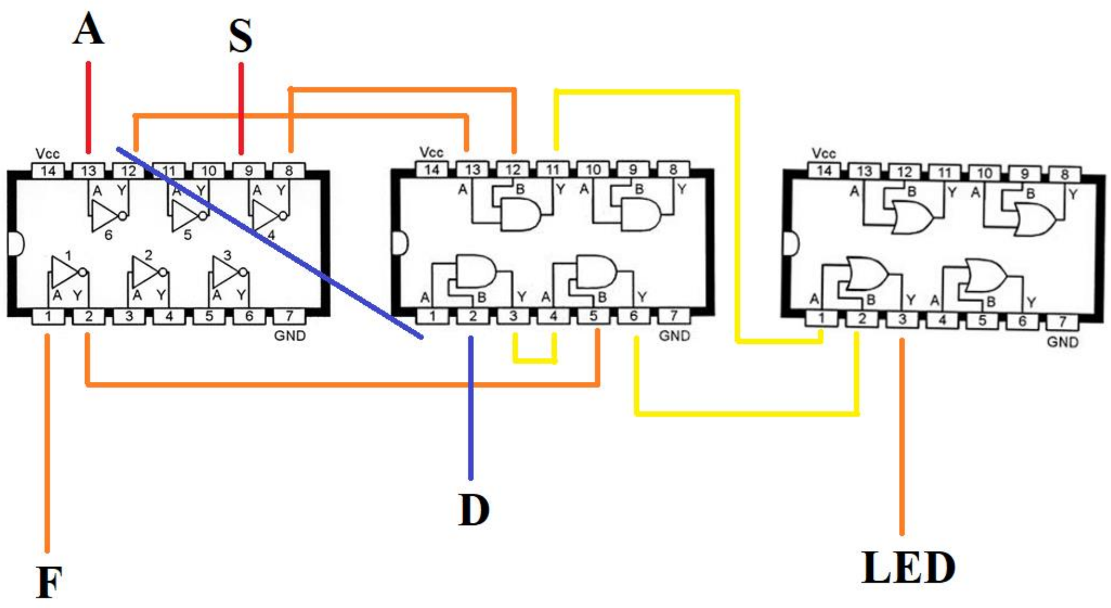
I designed my circuit on two breadboards, one for the AND/OR/INVERTER gates and one for the NAND gate circuit. In total I used one AND, one NOT and one OR chip for the first one and two NAND chips for the second one. This can be done as there are more than one logic gates in a chip.

The positive supply was DC which was set to 3V according to the Soulbay power supply. I connected a 1 kΩ resistor to the positive side of the power supply to further limit the current passing. As seen below, the red jumper wires connect to the voltage supply of each chip and the yellow jumper wires connect to the ground of each chip. Moreover, I have also made the rails common by connecting the positive-positive and negative-negative buses through jumper wires.

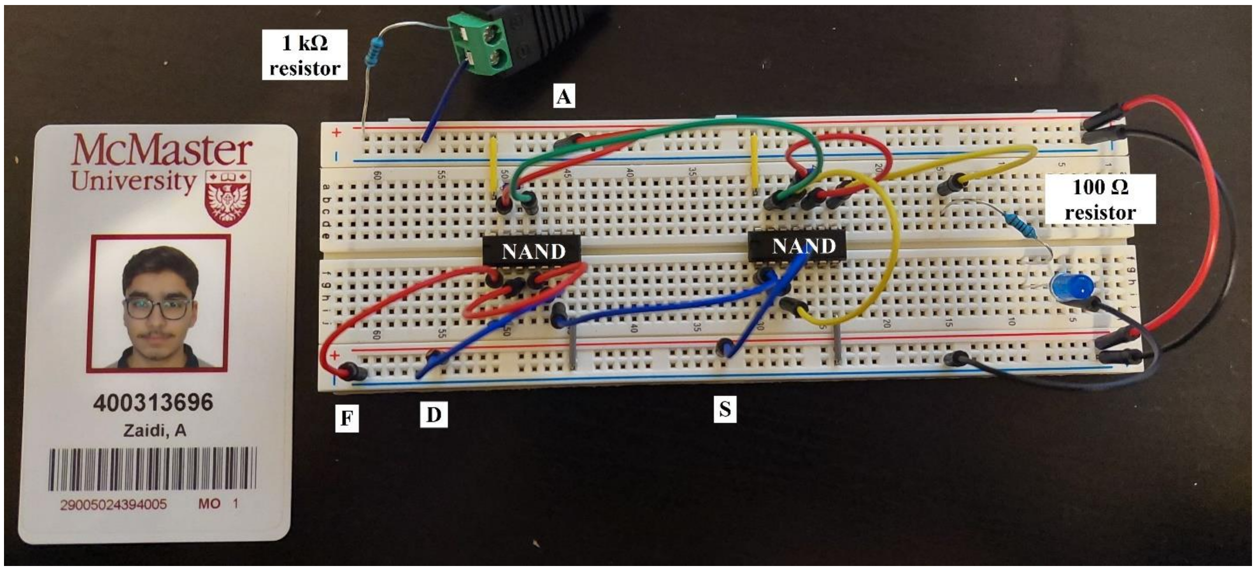
Below is the circuit for my AND/OR/INVERTER gate implementation.

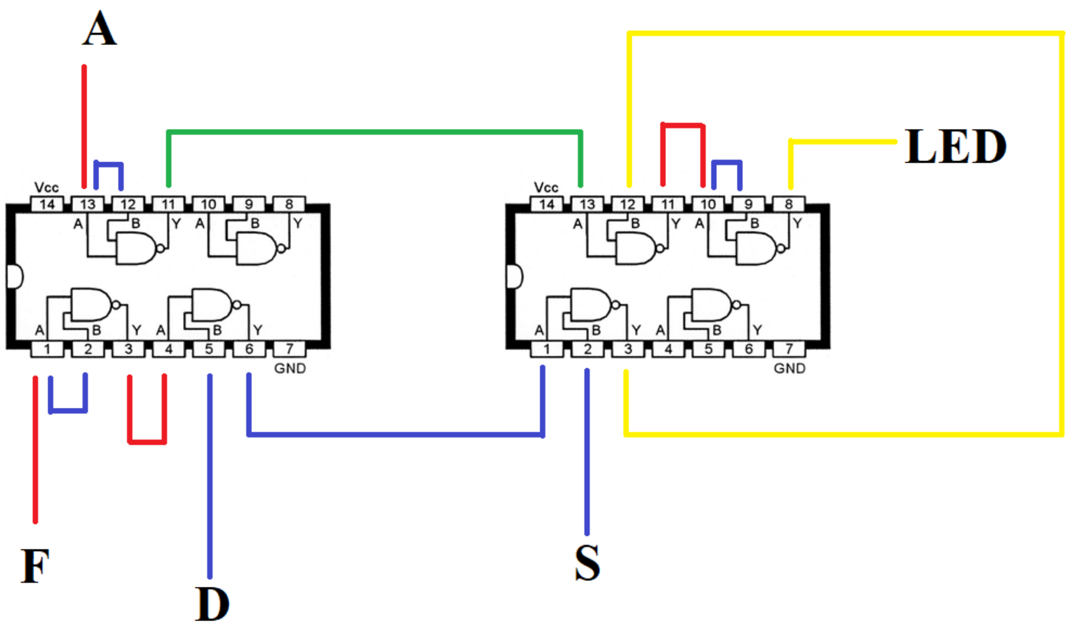


I also made a diagram highlighting the connections as they are a little bit hard to see in the above picture. The wires are coloured according to the colours in the picture above.

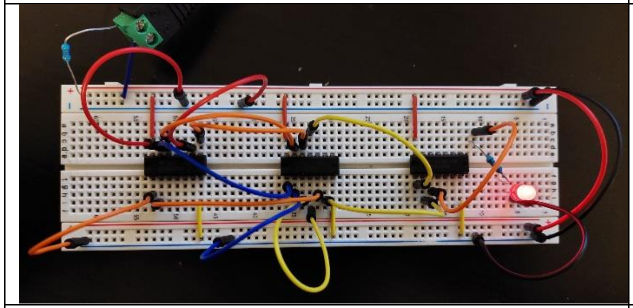
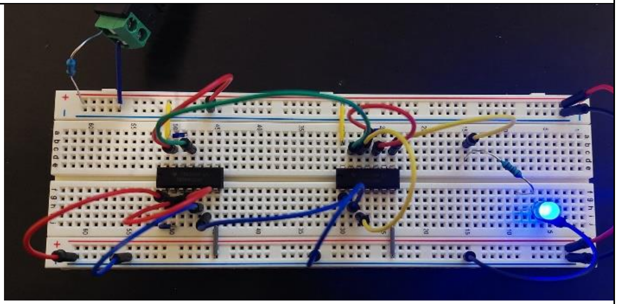
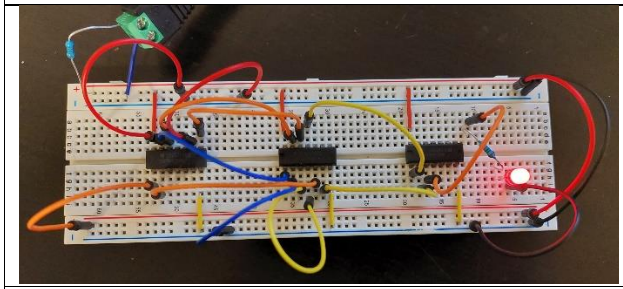
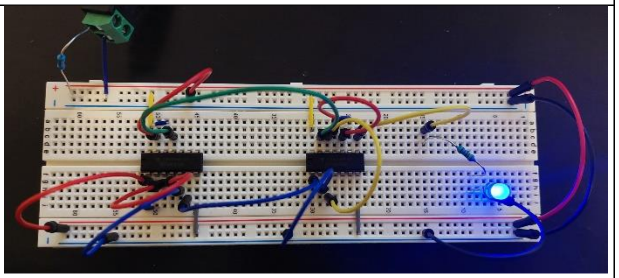


I then repeated the same solution but now with only NAND gates.

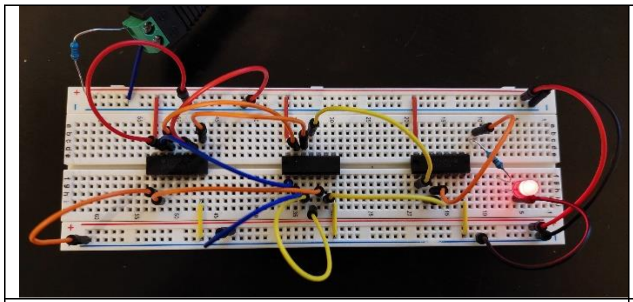
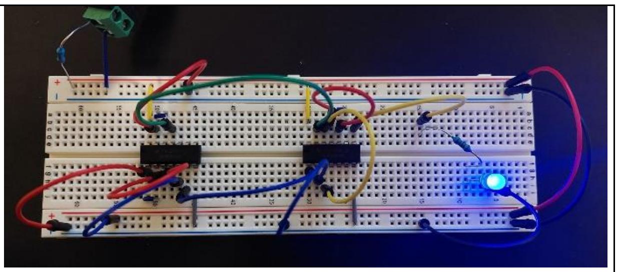
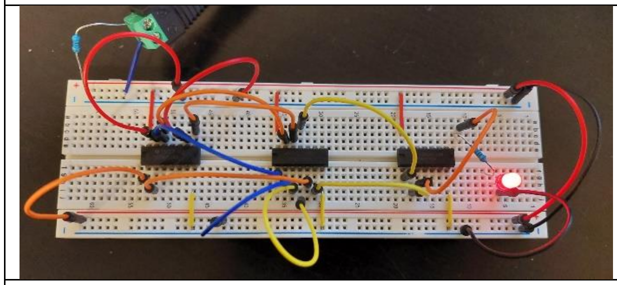
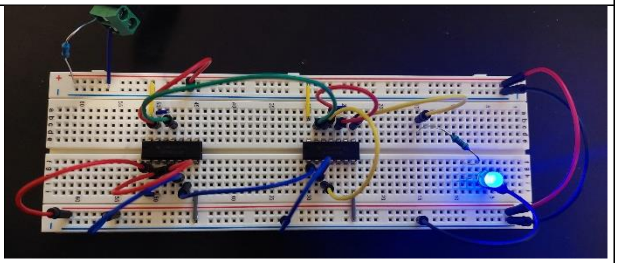
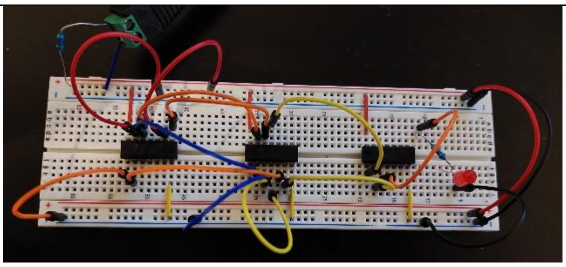
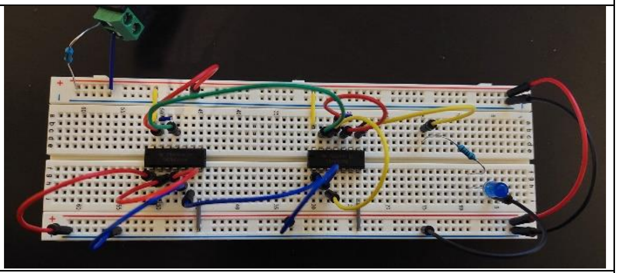
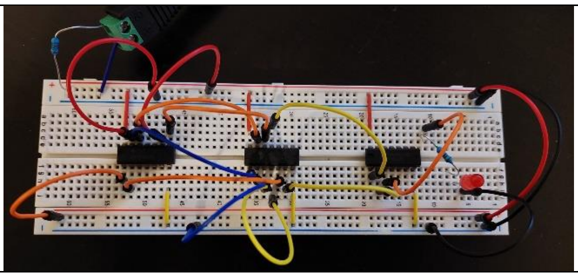
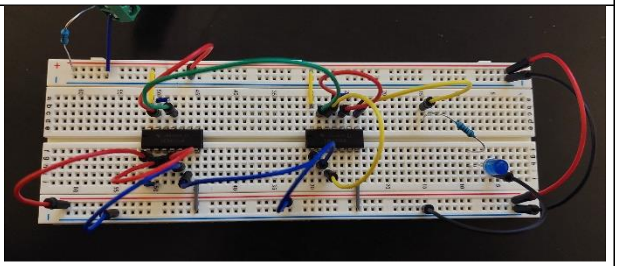
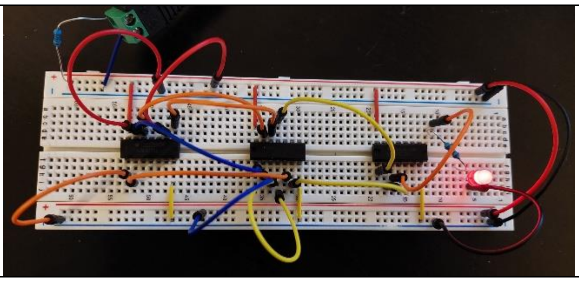
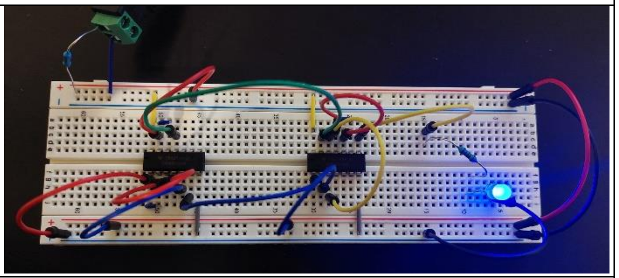




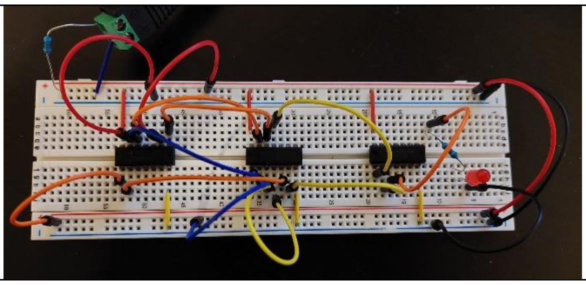
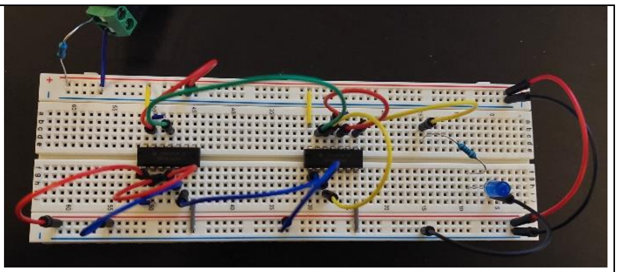
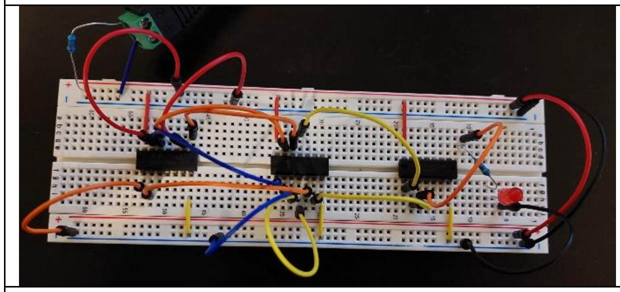
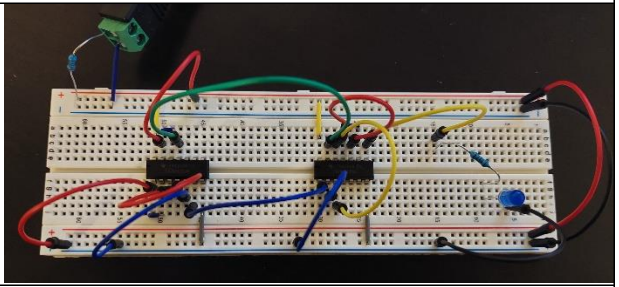
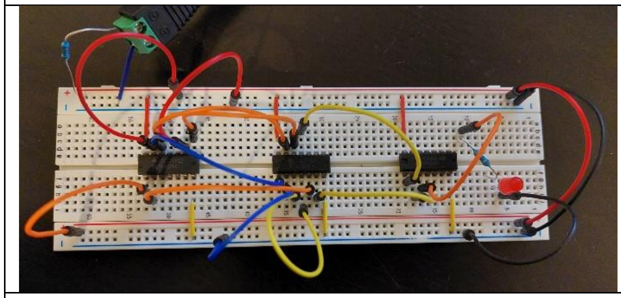
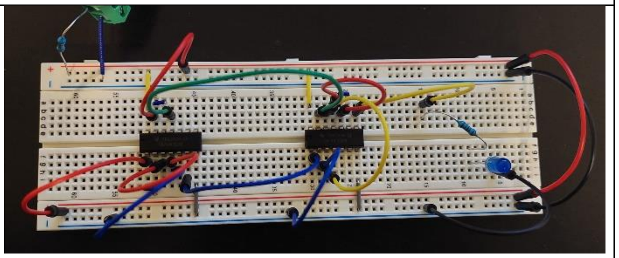
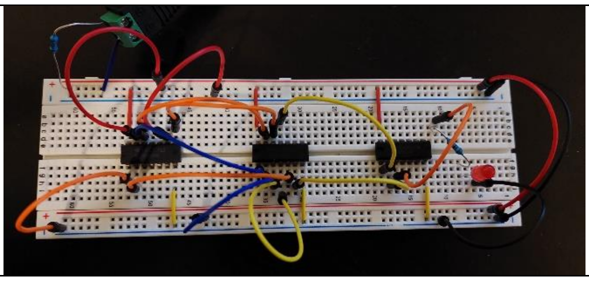
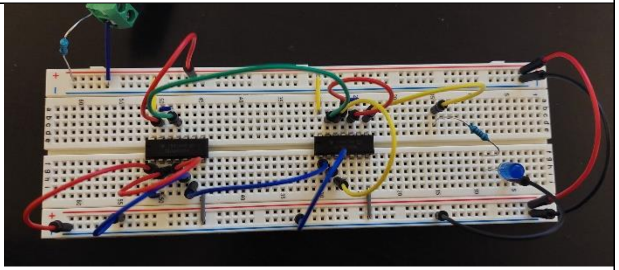
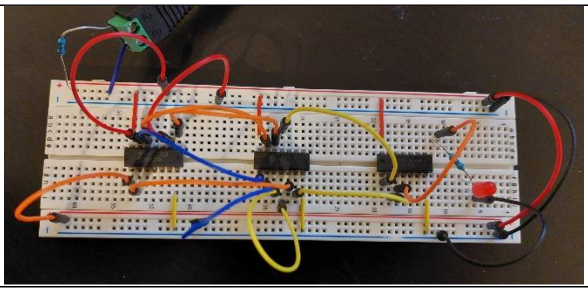
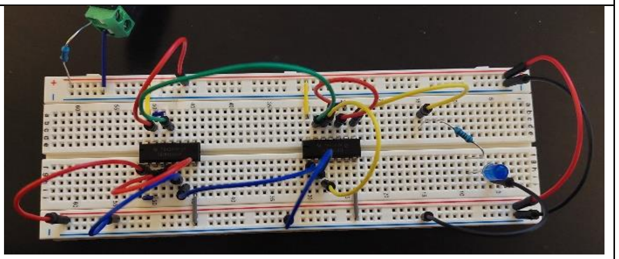
Now with the circuits made, I shall apply a range of inputs (from 0 to 15) and then check the output based on the state of the LED (on means TRUE and off means FALSE).

AND/OR/INVERTER	NAND
0000	0000
	
0001	0001
	
0010	0010

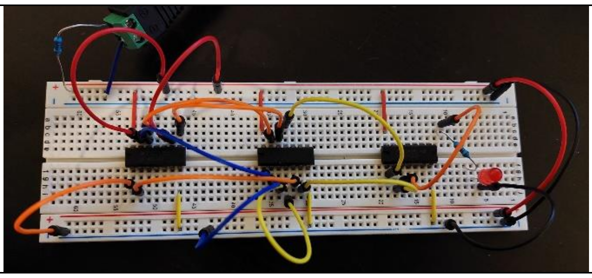
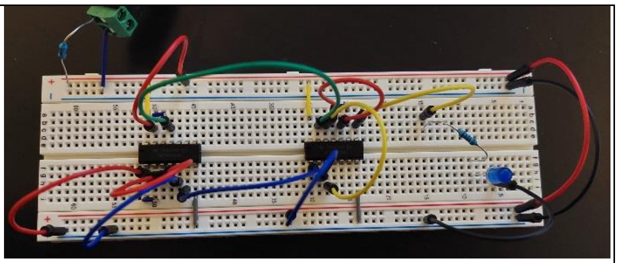
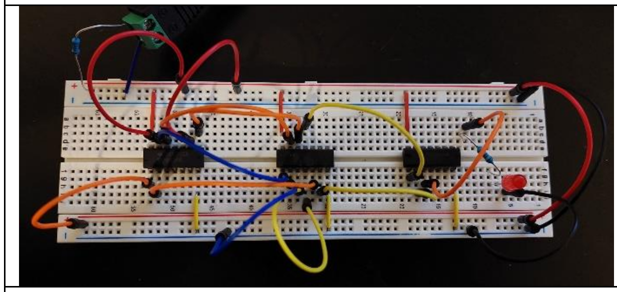
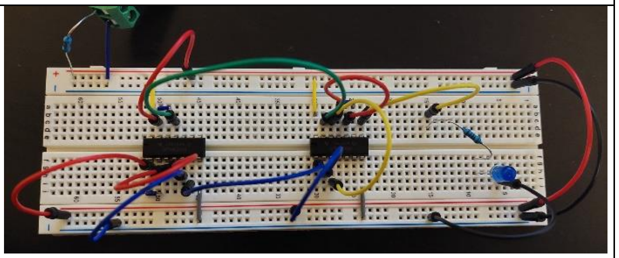
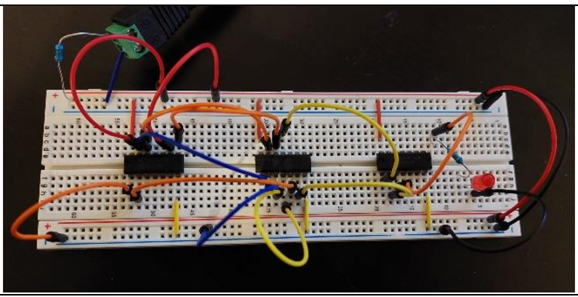
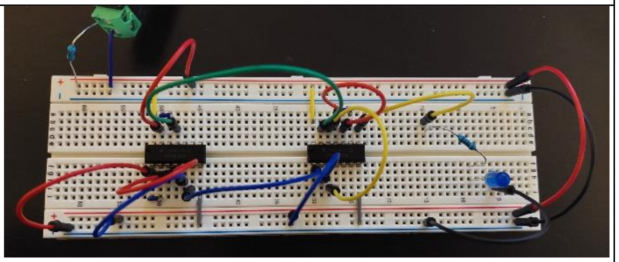
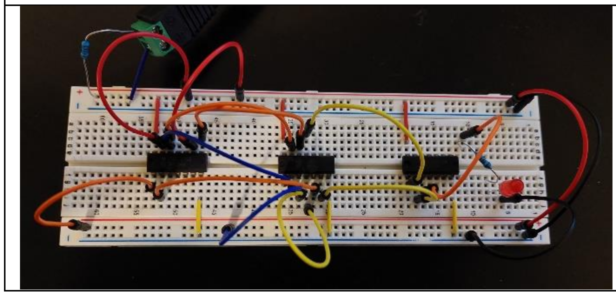
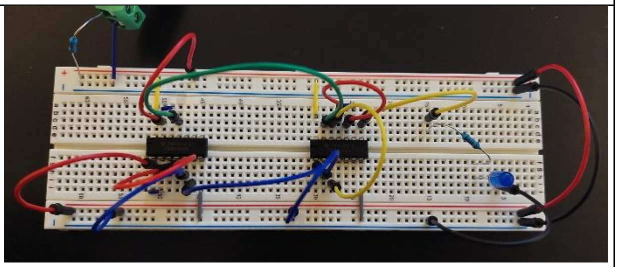
Syed Ali Zaidi
zaidis33

	
0011	0011
	
0100	0100
	
0101	0101
	
0110	0110
	
0111	0111

Syed Ali Zaidi
zaidis33

	
1000	1000
	
1001	1001
	
1010	1010
	
1011	1011
	
1100	1100

Syed Ali Zaidi
zaidis33

	
1101	1101
	
1110	1110
	
1111	1111
	

These results can be summarized into a table:

Decimal Number	A	S	D	F	Physical Output
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	0
15	1	1	1	1	0

The table shows that the outputs are the same when compared to the analogue and physical solution (see Discussion below). This is easier to achieve in digital circuits as there is no uncertainty seen in the measurements; it is only measured in states. The states also has a wide range of voltages where it is on and off thus the error is not present which allows us to achieve an ideal solution from the physical circuit.

Discussion

As seen below, the results I obtained from the three solutions all agree with each other. The one thing to note is that the analytical solution contains X in place of the ignored values (as they were not in the range) whereas they were included in the digital and physical solutions for completion.

In my digital and physical solution, I chose the sum of products method to implement the NAND gate circuit. This was because the result from that implementation was simpler causing me to easily replicate it. The results from both the implementations were the same which is proven by the truth table of the product of sums which is the same as the sum of products.

Syed Ali Zaidi
zaidis33

In my opinion, both the AND/OR/INVERTER and NAND gates method had both pros and cons. The AND/OR/INVERTER method had very easy derivation of the expression which led to easy implementation. The circuit could be divided into parts where it was known what each logic gate is doing. However, as seen in my physical solution, this type of circuit required more chips than the NAND gate which led to more wiring and a more difficult interpretation of results physically.

The NAND gate, on the other hand, required less wiring and it also uses less transistors which allow us to save on costs which might be very useful if working for a company on a large-scale project. I also found the NAND gate implementation to be the most useful when replicating the circuit physically as it allows less mistakes to occur. However, it has a more difficult derivation which relies on the use of higher Boolean Algebra concepts like De Morgan's laws.

Analytical Solution				
AS/DF	00	01	11	10
00	X	1	1	1
01	0	0	0	1
11	0	0	X	0
10	0	0	0	0

Digital Solution				
AS/DF	00	01	11	10
00	1	1	1	1
01	0	0	0	1
11	0	0	0	0
10	0	0	0	0

Syed Ali Zaidi
zaidis33

Physical Solution				
AS/DF	00	01	11	10
00	1	1	1	1
01	0	0	0	1
11	0	0	0	0
10	0	0	0	0

Reflection

In conclusion, this lab was a very refreshing change of pace from the analogue circuits. I found its concepts to be very linked to computer science and engineering which I am finding very interesting. I am also making connections to this with my programming course which allows me to further strengthen my knowledge and apply it. Overall, I am finding it to be very relevant to my field of study especially in embedded systems. Its applications stretch from burglar alarms to digital electronics to forming the basic architecture in computers. Thus, by introducing us to form a basic circuit analytically, digitally, and physically and by exposing us to concepts like Boolean Algebra and K-mapping, our foundation for digital electronics has been built.