EC-273

Digital Circuit And Systems Lab
Experiment 3-4

Pranav Mittal(19095076)
Guguloth Yakesh(19095038)
Rahul Ponnala(19095073)

Experiment 3

AIM :-

To Verify the truth table of half Subtractor by using the ICs of XOR, NOT and AND gates and of full Subtractor by using the ICs of XOR, AND NOT and OR gates respectively and analyses the Working of half Subtractor and full Subtractor circuit with the help of LED's in Simulator 1 and verify the truth table only of half Subtractor and full Subtractor in Simulator 2.

Theory :-

Introduction :-

Subtractor circuits take two binary numbers as input and subtract one binary number input. Similar to adders. it gives out two outputs. difference and borrow (carry in the case of Adder). there are two types of Subtractor.

1) Half Subtractor.

2) full Subtractor.

1) Half Subtractor :-

The half subtractor is a Combinational Circuit which is used to perform subtraction of two bit. it has two inputs. A [minuend) and B [subtrahend] and two outputs difference and Borrow. the logic symbol and truth table are shown below.

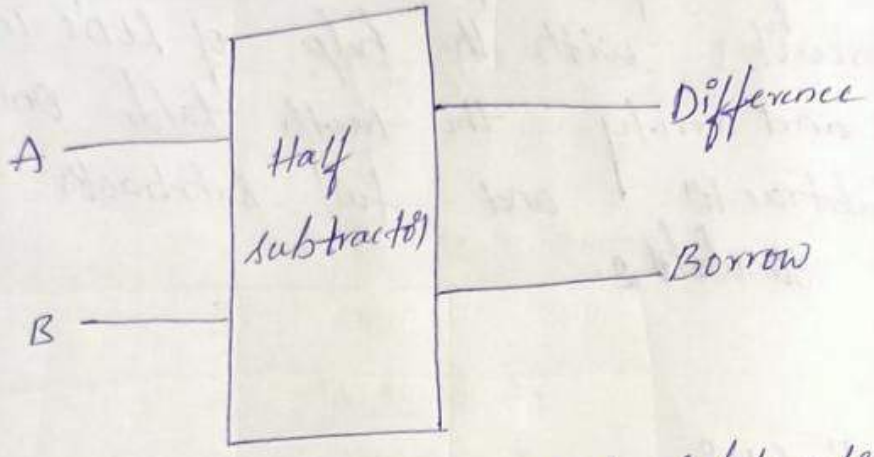
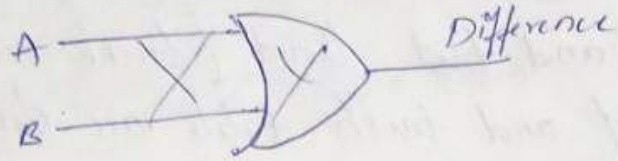


figure :1: logic symbol of Half subtractor.

Inputs		Outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

fig2: Truth table of Half subtractor.



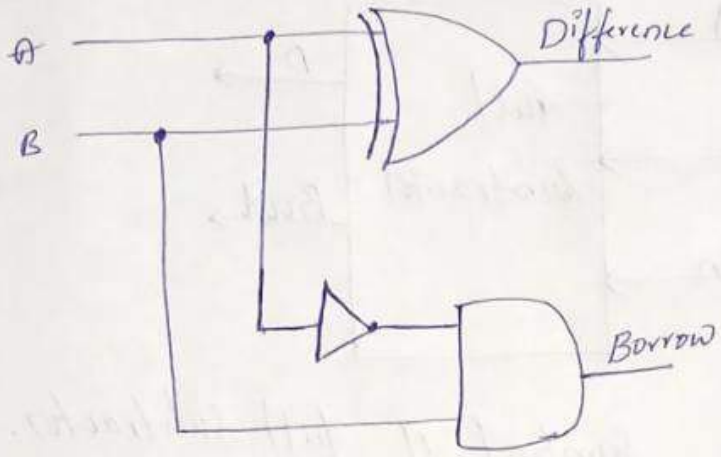


fig-3:- circuit diagram of Half Subtractor.

From the above truth table we can find the boolean expression

Difference - $A \oplus B$

Borrow - $A'B$.

from the equation we can draw the half-subtractor circuit as show in the figure 3.

2] Full Subtractor :-

A full subtractor is a combinational circuit that performs subtraction involving three bits. namely A [minuend] B [subtrahend], and Bin [borrow-in] it accepts three inputs A [minuend], B [subtrahend]

and Bin [borrow bit] and it produces two outputs D (difference) and but Bout [borrow out]. The logic symbol and truth table are shown below

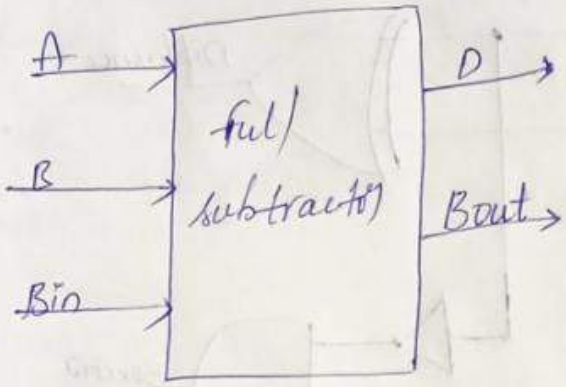


Fig 4:- logic symbol of full subtractor.

A	B	Bin	D	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Fig 5:- Truth table of full subtractor

from the above truth table we can find the boolean Expression

$D - A \oplus B \oplus Bin$

$Bout - A'Bin + A'B + B\,Bin$.

from the equation we can draw the full - Subtractor circuit as shown in the fig 6

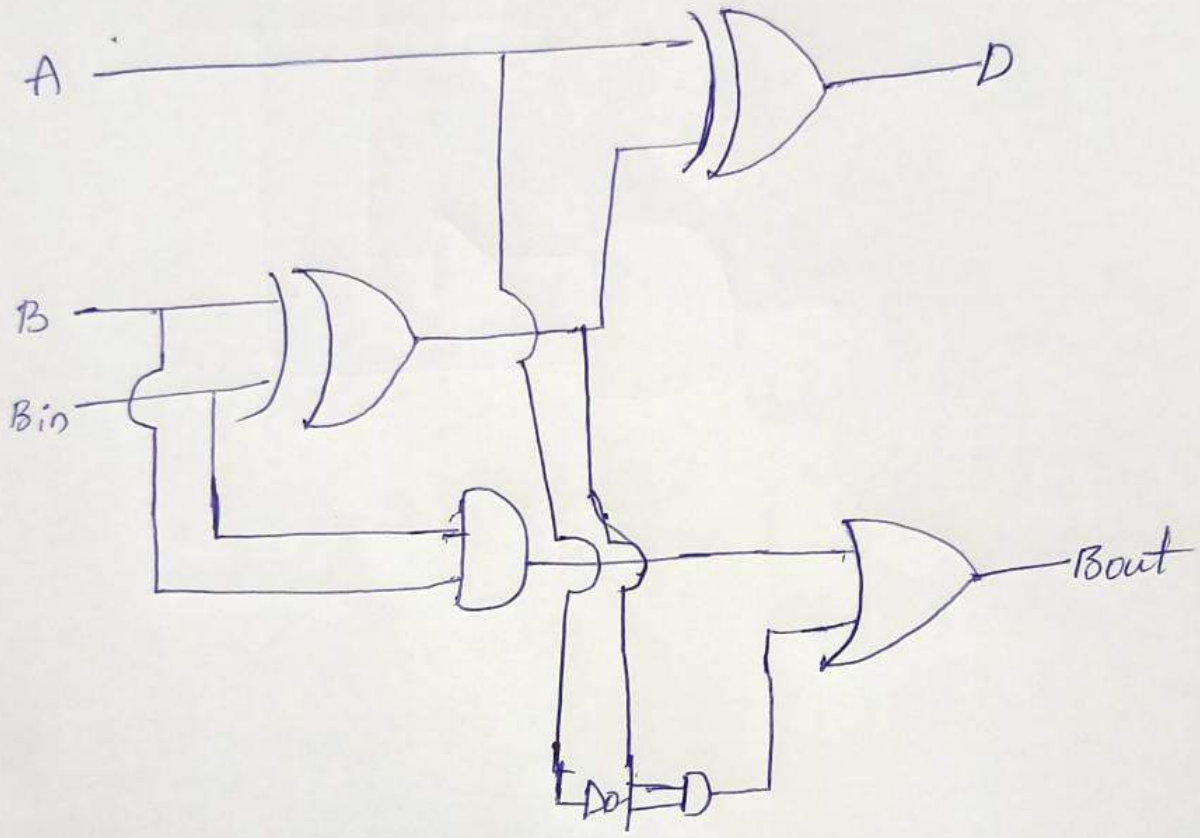
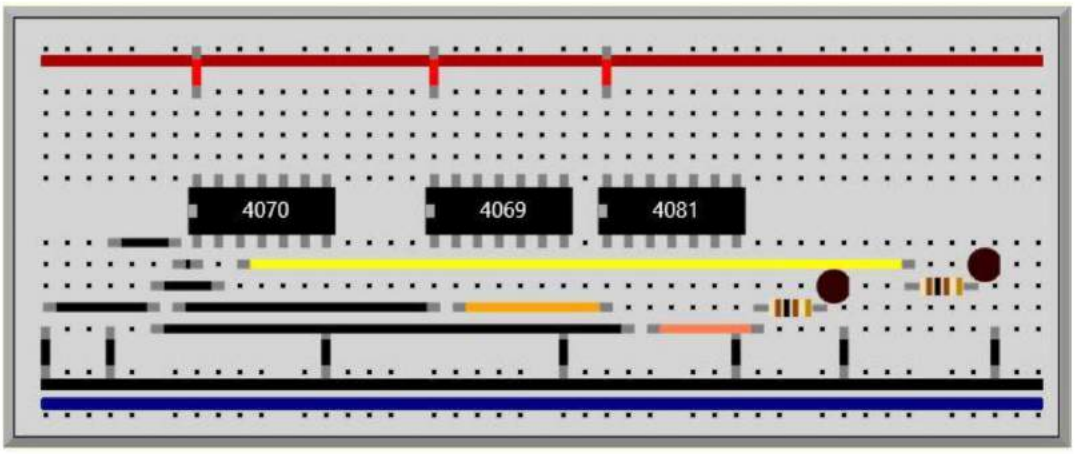


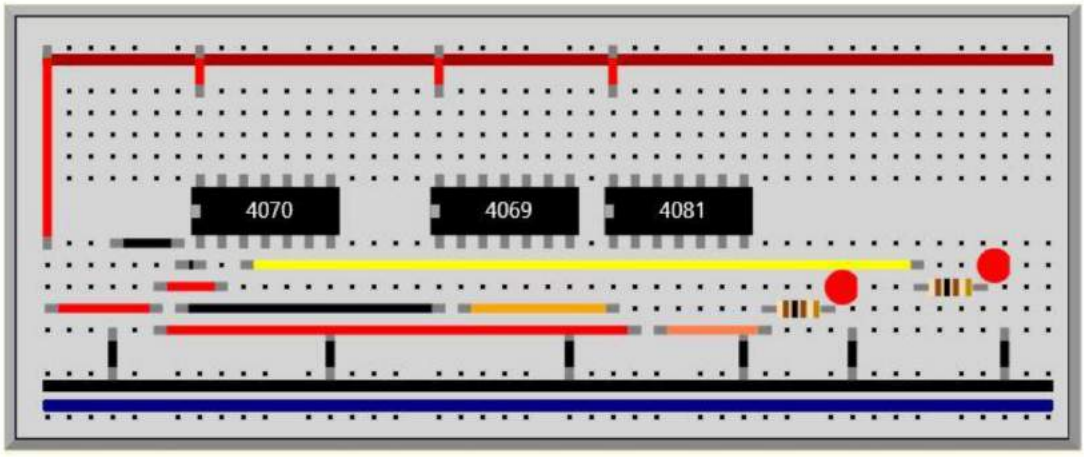
Fig 6: Circuit diagram of full Subtractor.

HALF SUBTRACTOR

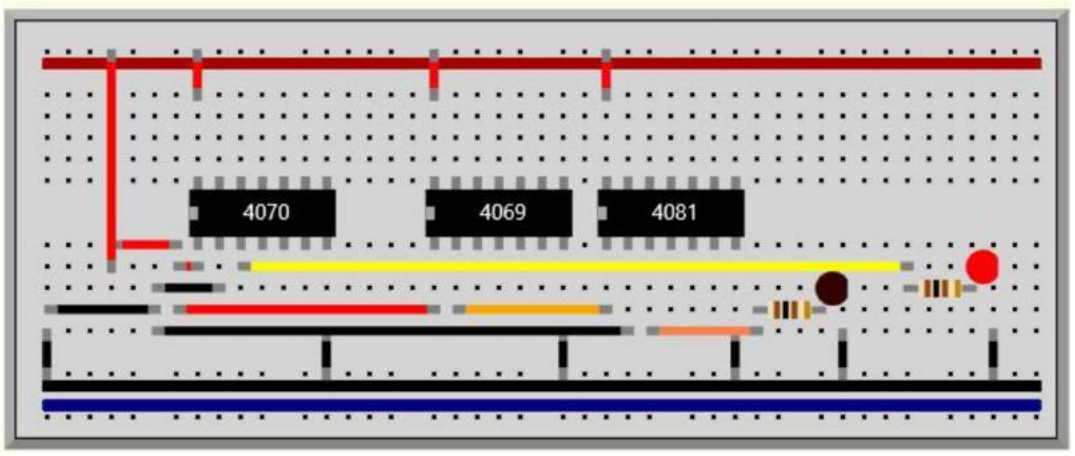
Input: A0 B0 Output:B0 D0



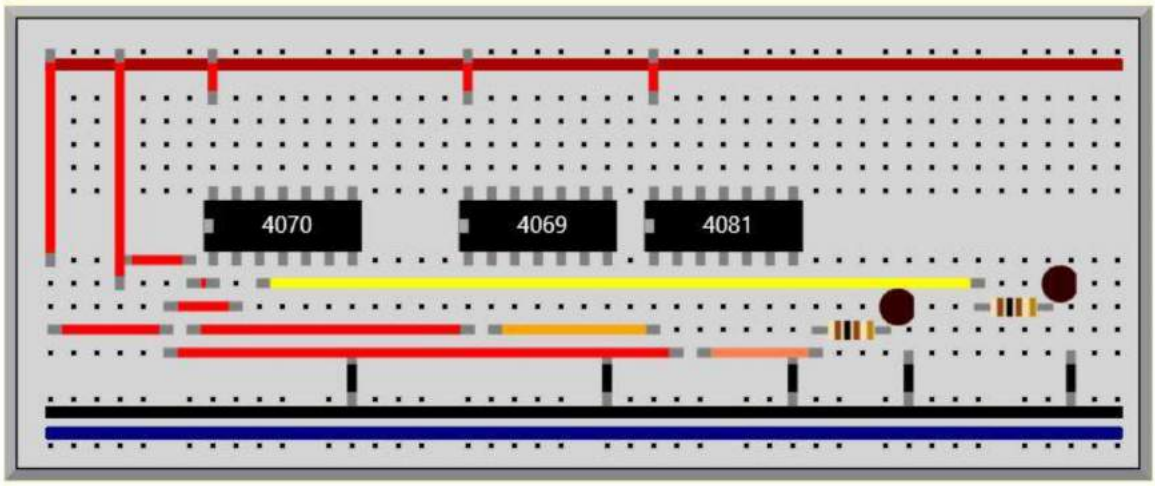
Input: A0 B1 Output:B1 D1



Input: A1 B0 Output:B0 D1

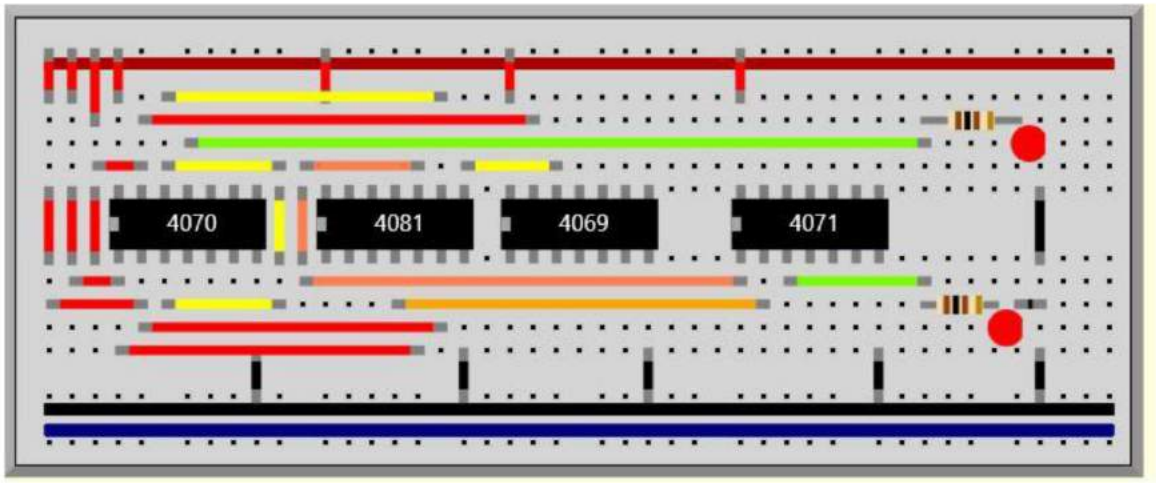


Input: A1 B1 Output:B0 D0

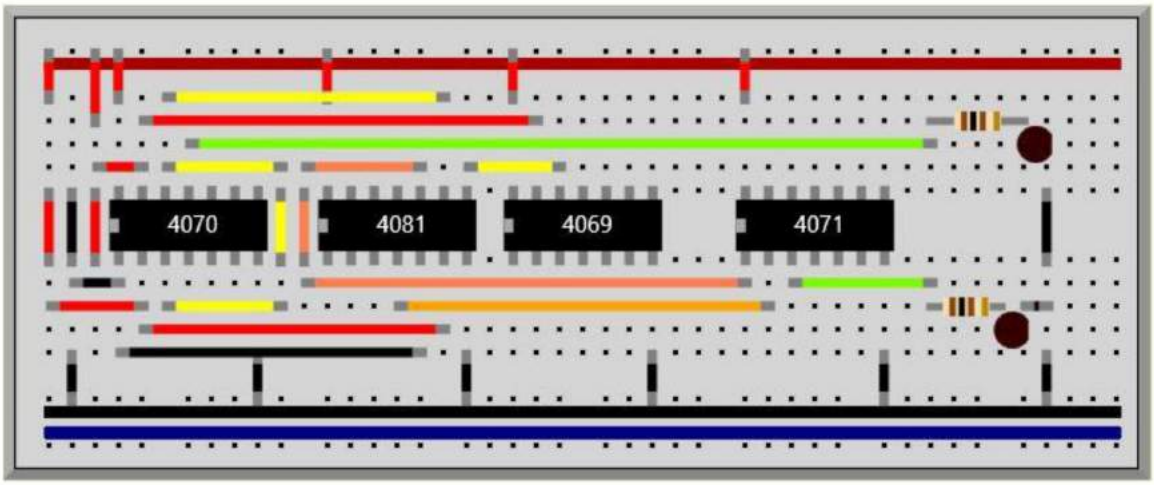


FULL SUBTRACTOR

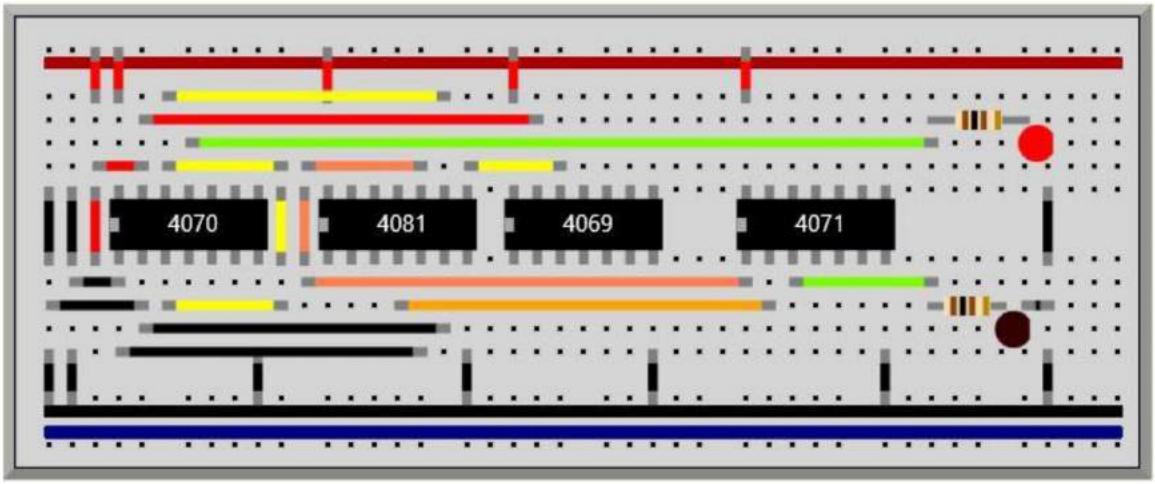
Input: A1 B1 Bin1 Output:Bout1 D1



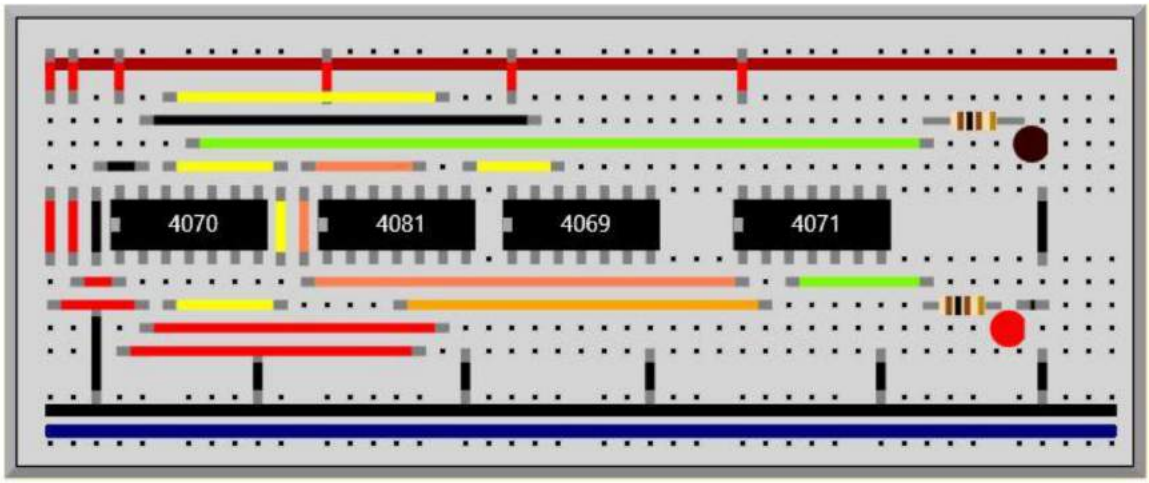
Input: A1 B0 Bin1 Output:Bout0 D0



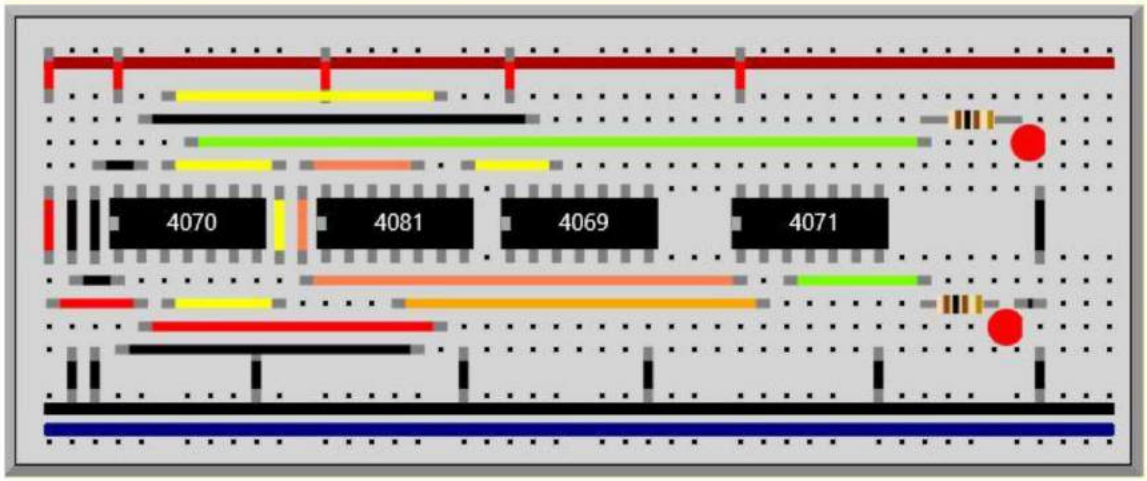
Input: A1 B0 Bin0 Output:Bout0 D1



Input: A0 B1 Bin1 Output:Bout1 D0



Input: A0 B1 Bin0 Output:Bout1 D1



Experiment 4

Aim:

To implement the logic functions i.e. AND, OR, NOT, Ex-OR, Ex-NOR and a logical expression with the help of NAND and NOR universal gates respectively

Theorey:

Logic gates are electronic circuits which perform logical functions on one or more inputs to produce one output. There are seven logic gates. When all the input combinations of a logic gate are written in a series and their corresponding outputs written along them then this input/output combination is called Truth table.

1.) Nand gate as Universal gate

NAND gate is actually a combination of two logic gates i.e. AND gate.

followed by NOT gate. So its output is complement of the output of an AND gate. This gate can have minimum two inputs. By using only NAND gates, we can realize all logic functions AND, OR, NOT, Ex-OR, Ex-NOR, NOR. So this gate is also called as universal gate.

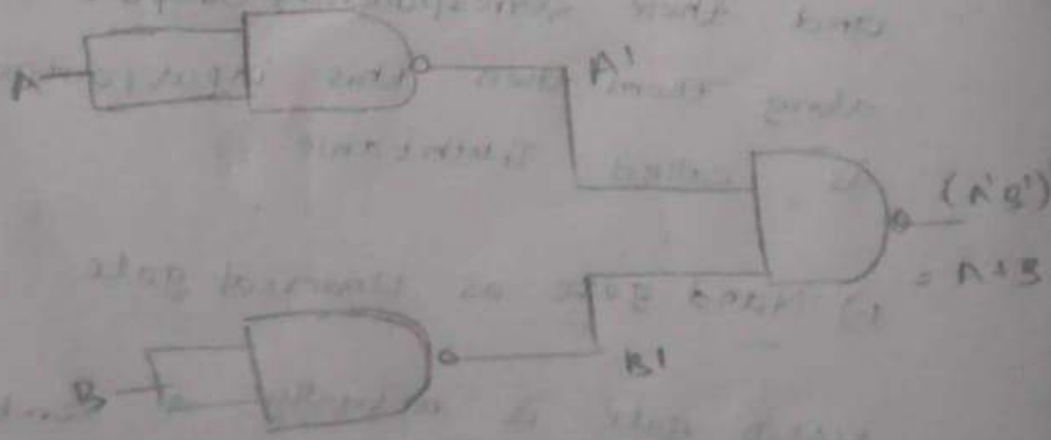
a.) NAND gates as OR gate

From DeMorgans theorems

$$(A \cdot B)' = A' + B'$$

$$(A' \cdot B')' = A'' + B'' = A + B$$

So give the inverted inputs to a NAND gate, obtain OR operation at output.



OR

NAND gates as OR gate

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

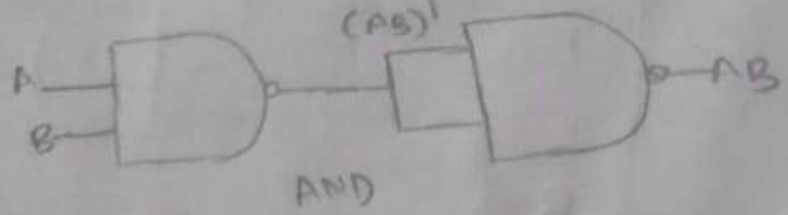
Truth table of OR

b.) NAND gates as AND gate

A NAND produces complement of AND gate. So, if the output of a NAND gate is inverted, overall output will be that of an AND gate.

$$Y = ((A \cdot B)')'$$

$$Y = (A \cdot B)$$



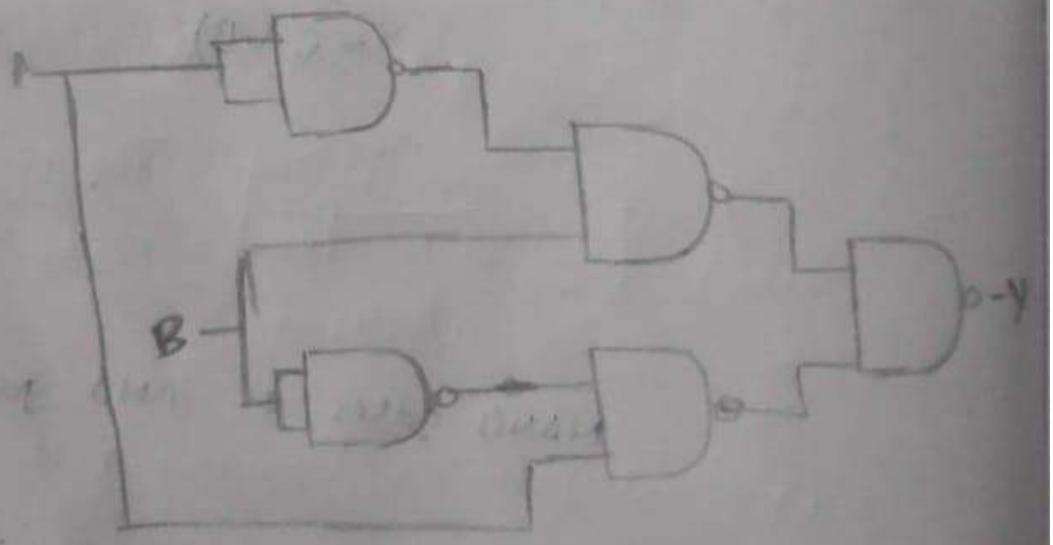
NAND gates as AND gate.

Input		output
A	B	Y=A·B
0	0	0
0	1	0
1	0	0
1	1	1

Truth table of AND.

c.) NAND gates as Ex-OR gate.

The output of a two input Ex-OR gate is shown by $Y = A'B + AB'$. This can be achieved with the logic diagram shown in the left side.



NAND gate as Ex-OR gate.

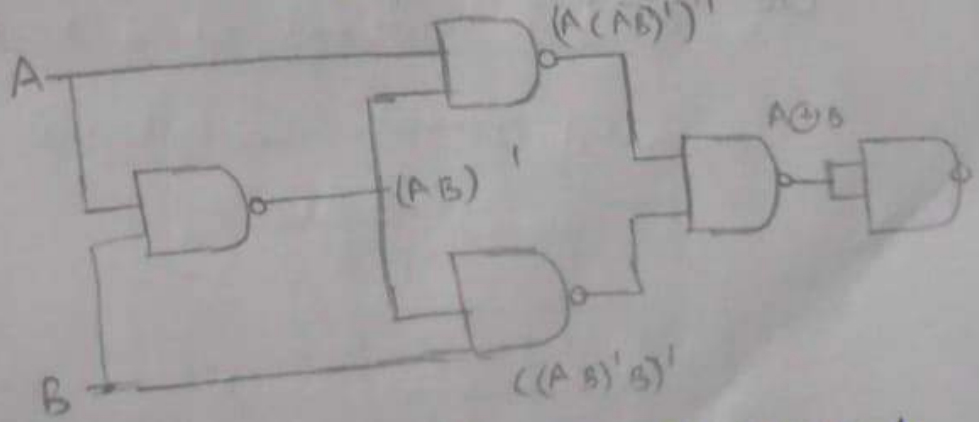
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Truth table of Ex-OR

d-) NAND gates as Ex-NOR gate

Ex-NOR gate is actually Ex-OR gate followed by NOT gate. So give the output of Ex-OR gate to a NOT gate, overall input is that of an Ex-NOR gate.

$Y = AB + A'B'$



NAND gates as Ex-NOR gate

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Truth table of Ex-NOR

e.) Implementing the simplified function with NAND gates only

We can now start constructing the circuit. First note that the entire expression is inverted and we have three terms ANDed. This means that we must

2) NOR gate is Universal gate

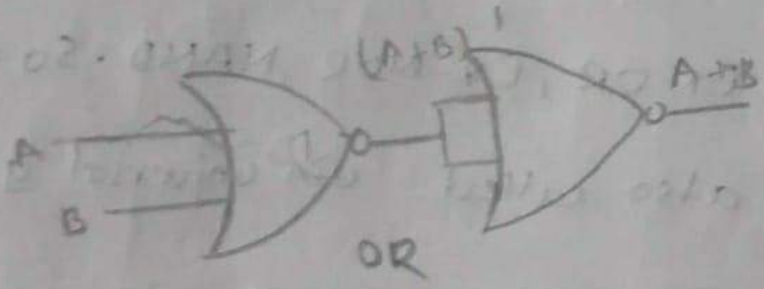
NOR gate is actually a combination of two logic gates, OR gate followed by NOT gate. So its output of an OR gate. This gate can have minimum two inputs, output is always one. By using only NOR gates, we can realize all logic functions. AND, OR, NOT, Ex-OR, Ex-NOR, NAND. So this gate is also called ~~a~~ universal gate.

a.) NOR gates as OR gate

A NOR produces complement of OR gate, So if the output of a NOR gate is inverted, overall output will be that of an OR gate.

$$Y = ((A+B)')'$$

$$Y = (A+B)$$



NOR gates as OR gate.

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

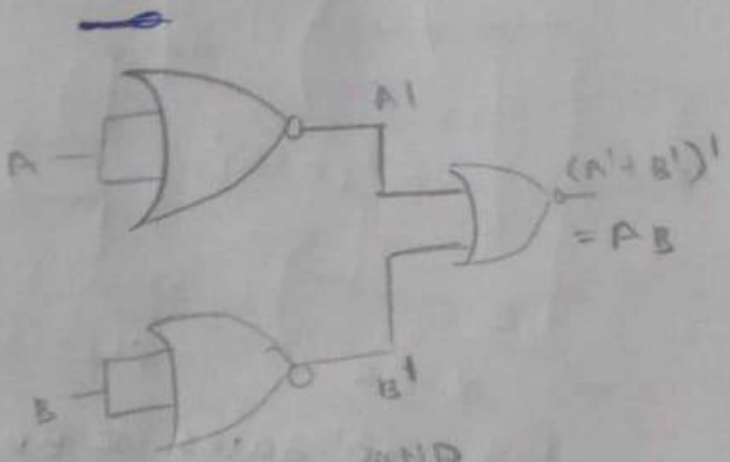
b) NOR gates as AND gate

From DeMorgan's theorems

$$(A+B)' = A'B'$$

$$(A'+B')' = A'B' + AB$$

So, give the inverted outputs to a NOR gate, obtain AND operation at output



AND

NOR gates as AND gate.

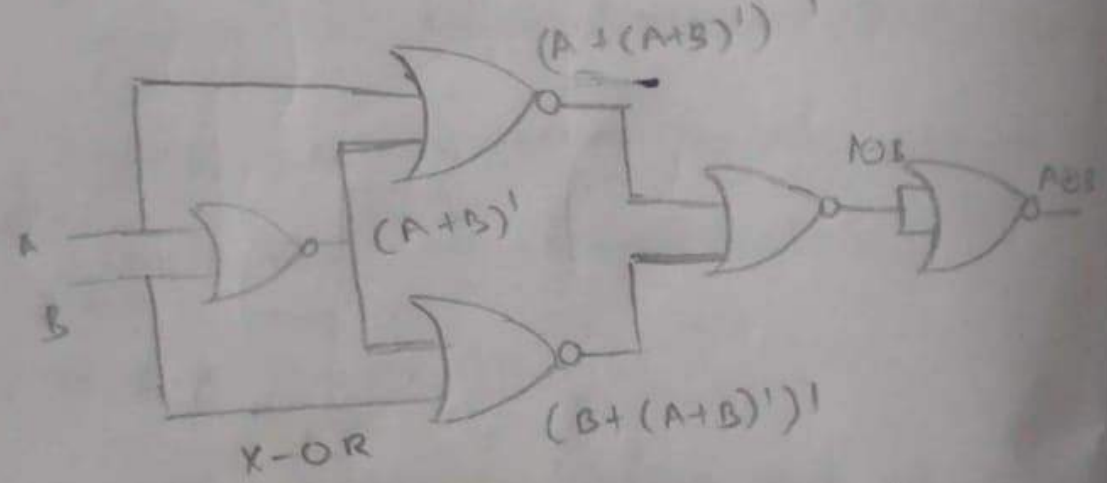
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Truth table of Ex-OR

(iv) NOR gates as Ex-OR gate

Ex-OR gate is actually Ex-NOR gate following by NOT gate. So give the output of Ex-NOR gate to a NOT gate, overall output is that of an Ex-OR gate.

$$Y = A'B + AB'$$



NOR gates as Ex-OR gate

A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Truth table of Ex-OR

d.) NOR gates as Ex-NOR gate

The output of a two input Ex-NOR gate is shown by $Y = AB + A'B'$. This can be achieved with the logic diagram shown in the left

Gate No.	Inputs	output
1	$A \cdot B$	$(A+B)'$
2	$A \cdot (A+B)'$	$(A+(A+B)')'$
3	$(A+B)' \cdot B$	$(B+(A+B)')'$
4	$(A+(A+B)')'$, $(B+(A+B)')'$	$AB + A'B'$

Now the output from gate no 4 is overall input of the configuration

$$Y = \left((A+(A+B)')' \; (B+(A+B)')' \right)'$$

$$Y = A \cdot (A+B)'' \cdot (B+(A+B)')''$$

$$= (A+(A+B)')'' \cdot (B+(A+B)')''$$

$$= (A+(A+B)')(B+(A+B)')$$

$$= (A+A'B')(B+A'B')$$

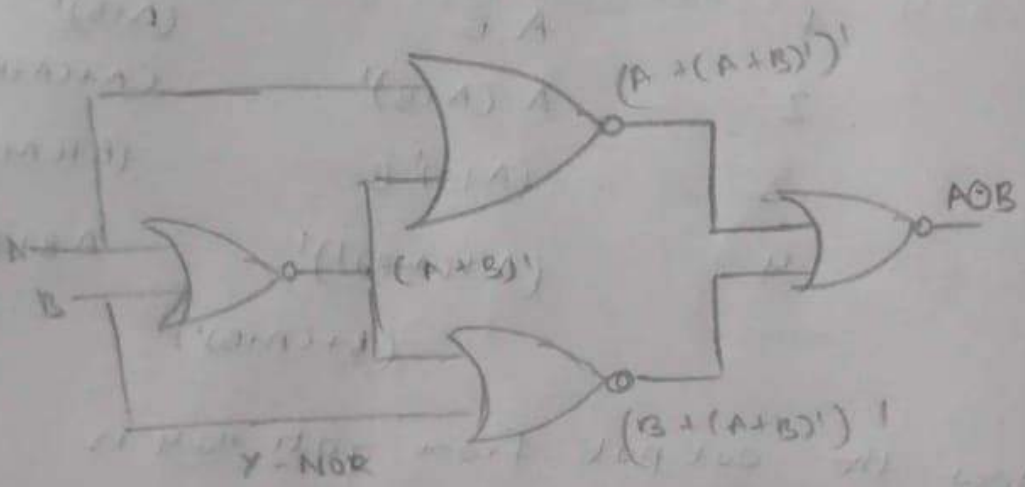
$$= (A+A') \cdot (A+B')(B+A')(B+B')$$

$$= 1 \cdot (A+B') \cdot (B+A') \cdot 1$$

$$= A(B+A') + B'(B+A')$$

$$= AB + 0 + 0 + B'A'$$

$$\Rightarrow \boxed{Y = AB + A'B}$$



NOR gates as Ex-NOR gate

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Truth Table of NOR gate only.

(09) a) Constructing a circuit with NOR gates only

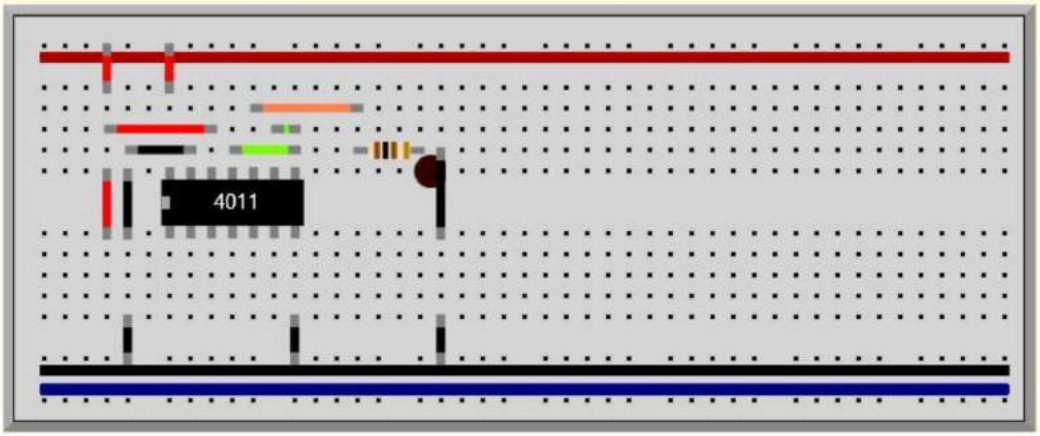
Designing a circuit with NOR gates only uses the same basic techniques as designing the a circuit with NAND gates, that is the application of demorgan's theorem. The only difference between NOR gate design and NAND gate design is the former must eliminate product terms and the later must be eliminate sum terms.

$$F = (((C \cdot B' \cdot A) + (D \cdot C' \cdot A) + (C \cdot B' \cdot A))')'$$

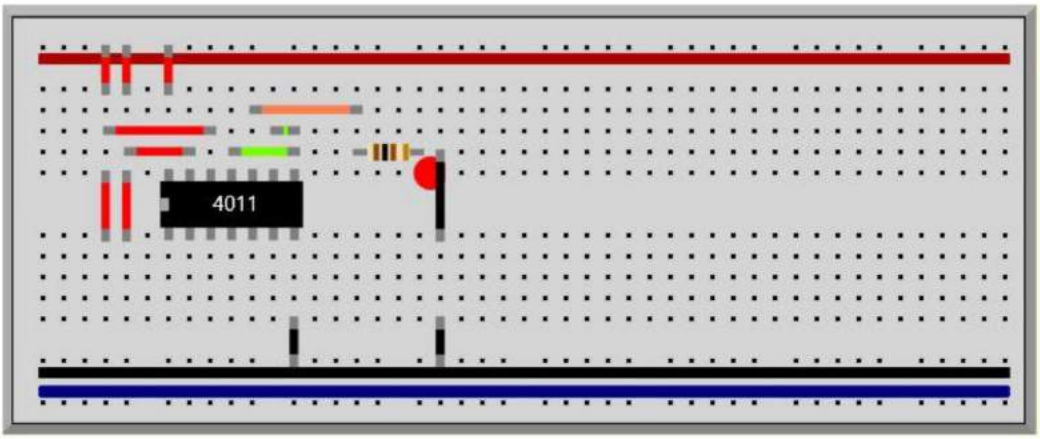
USING NAND

AND

Input:1 0 Output:0

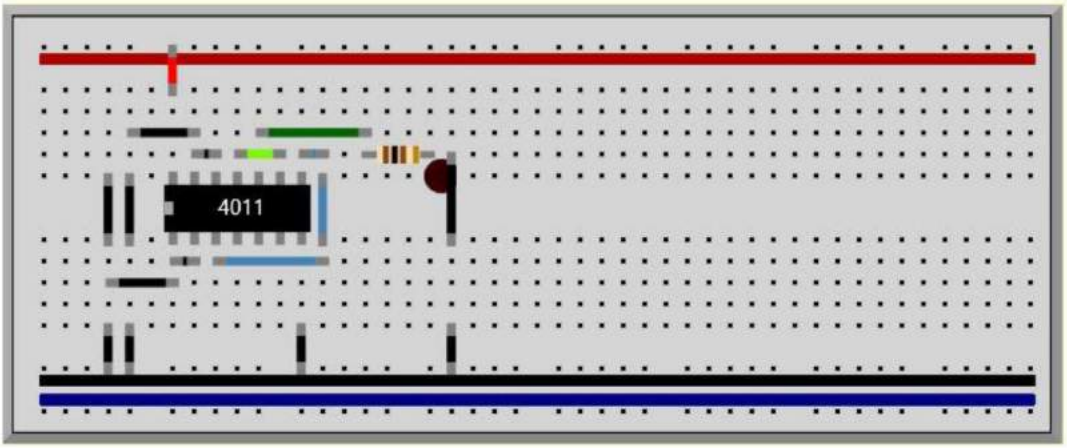


Input : 1 1 Output:1

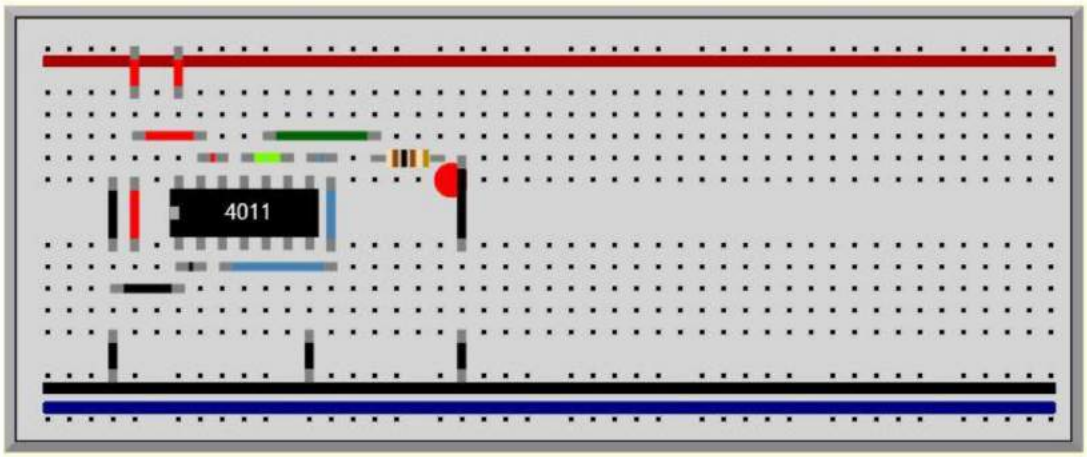


OR Gate

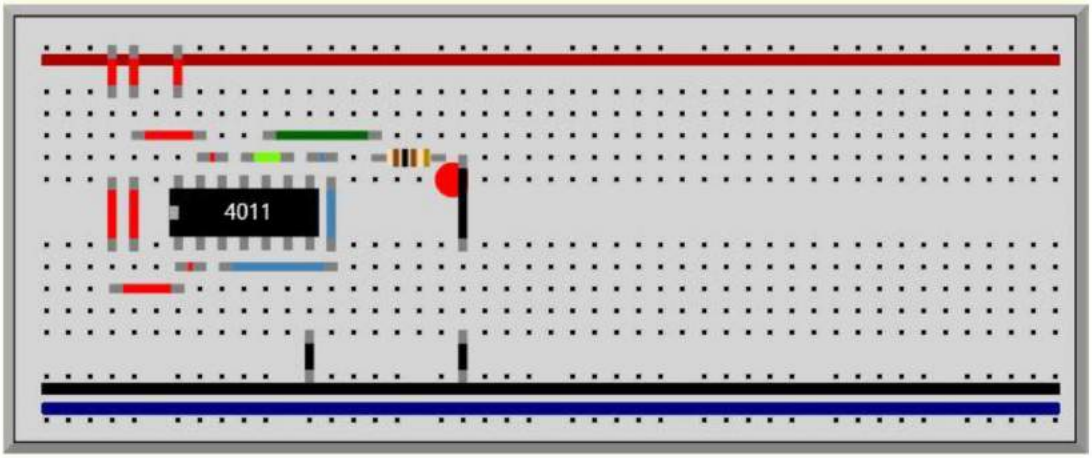
Input:0 0 Output:0



Input:1 0 Output:1

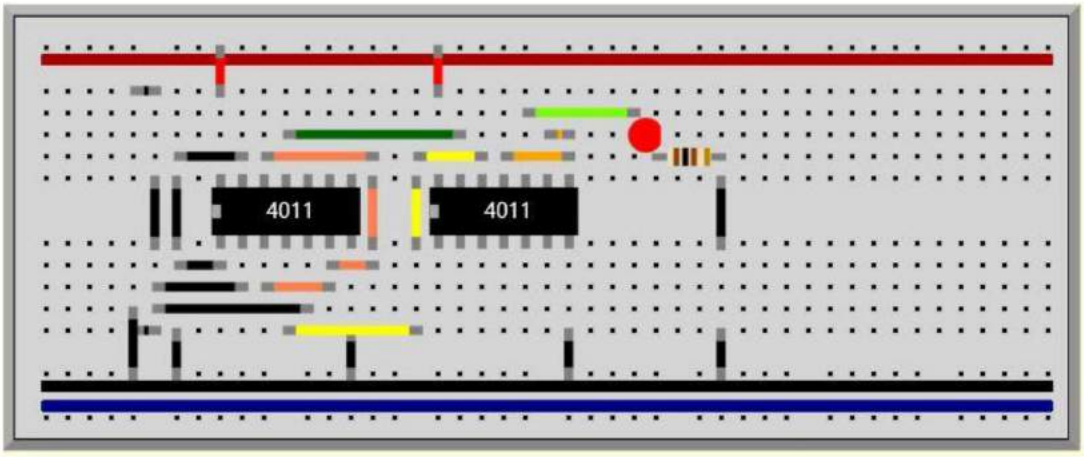


Input:1 1 Output:1

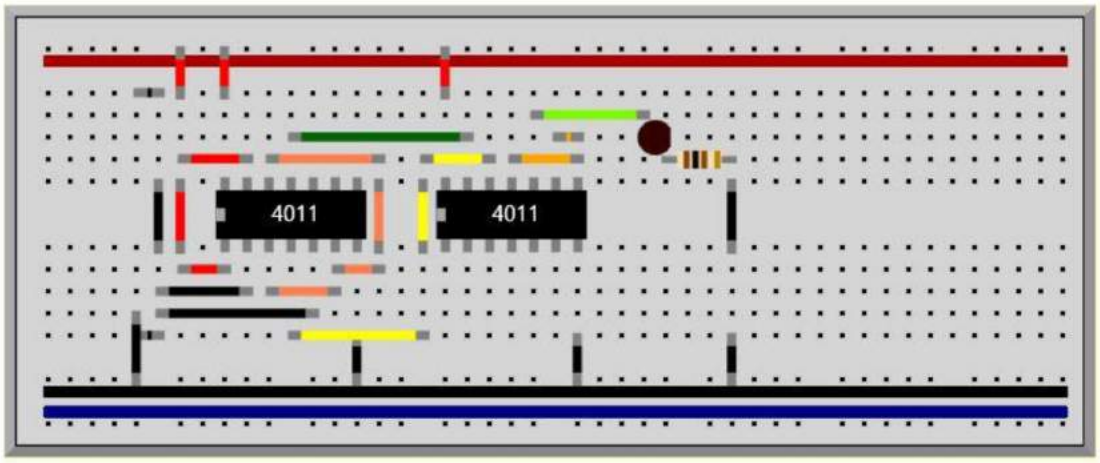


XNOR

Input:0 0 Output:1

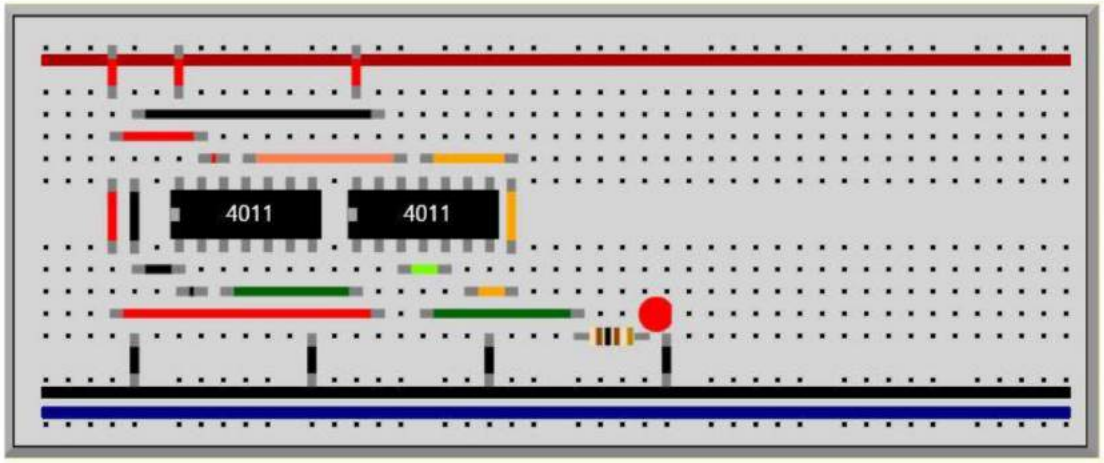


Input:1 0 Output:0

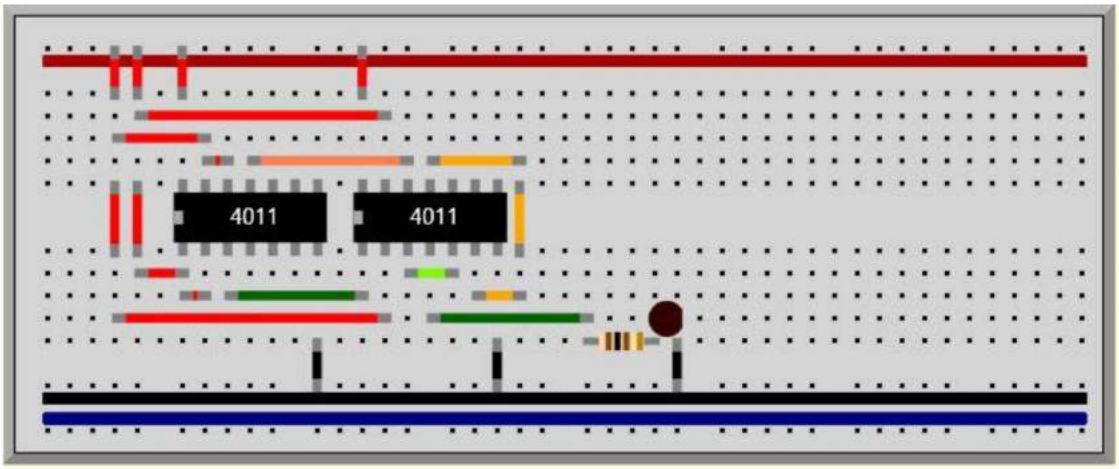


XOR

Input:1 0 Output:1



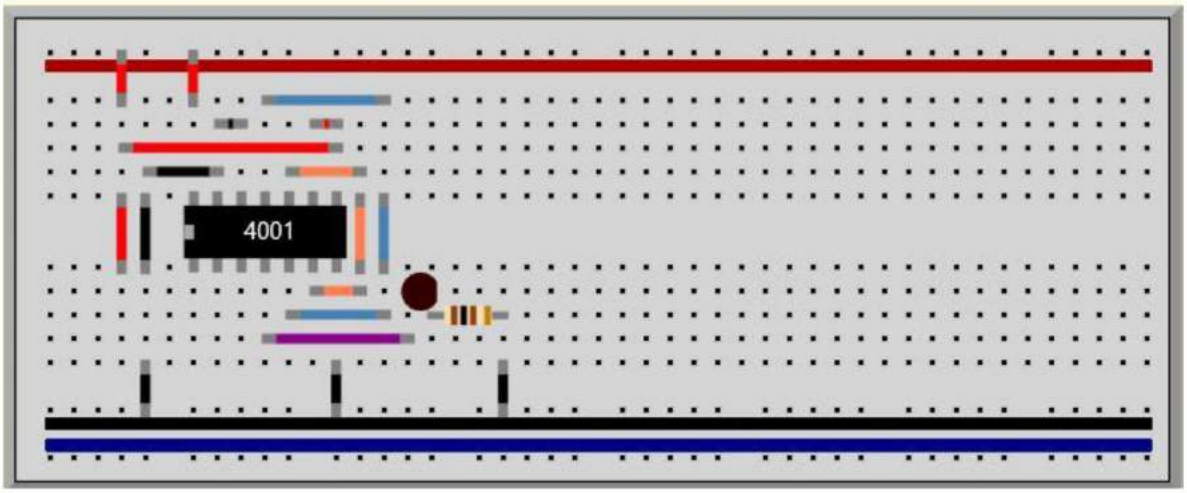
Input:1 1 Output:0



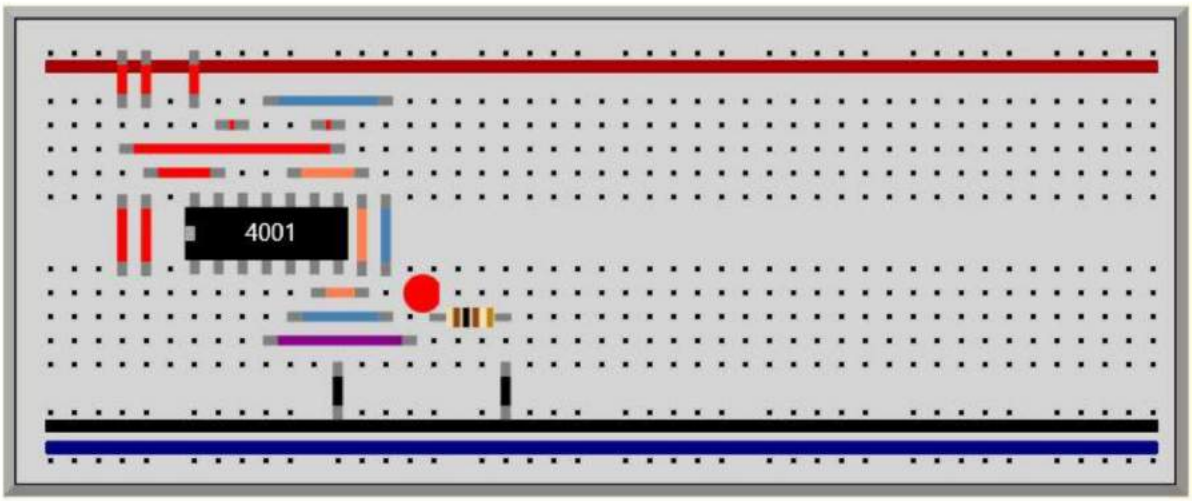
USING NOR GATE

AND

Input:1 0 Output:0

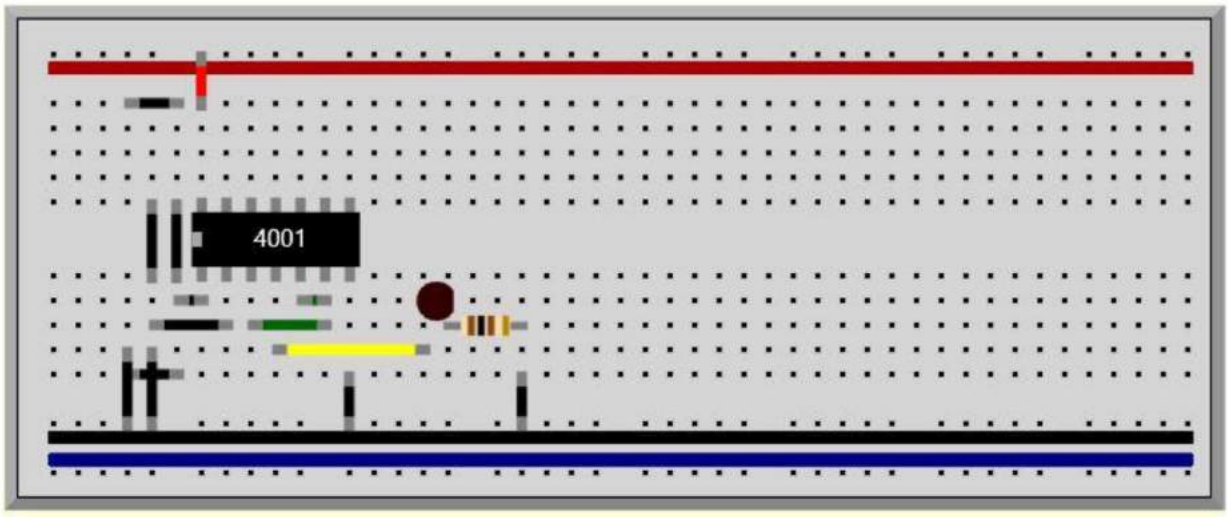


Input:1 1 Output:1

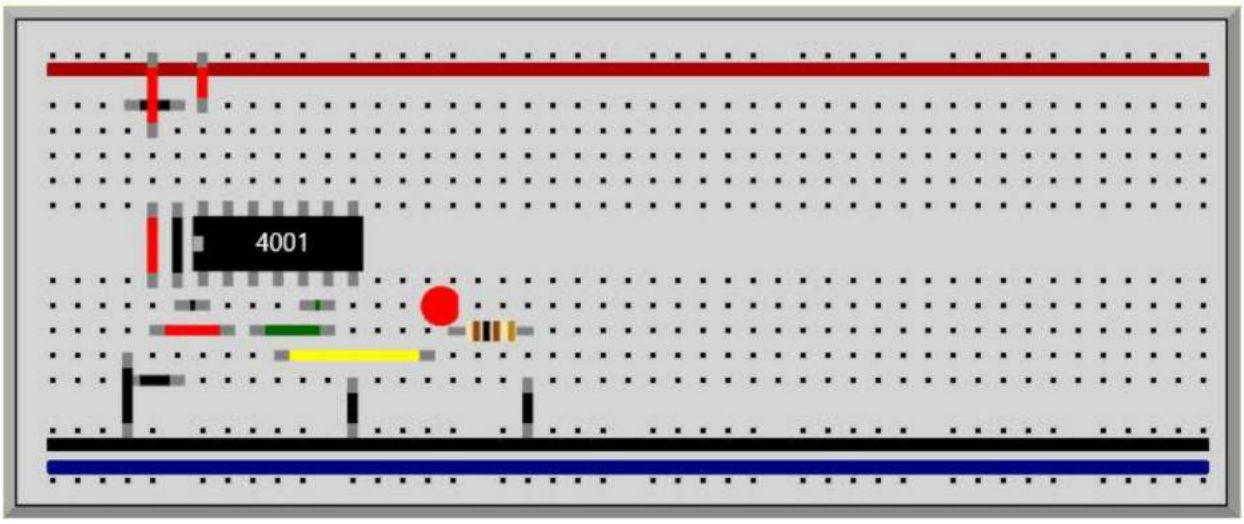


OR

Input:0 0 Output:0

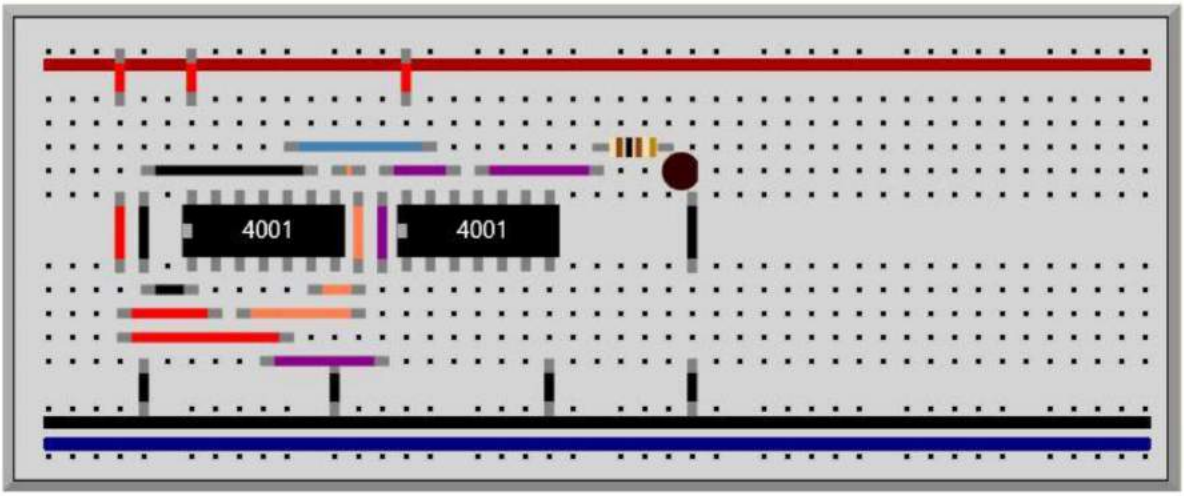


Input:1 0 Output:1

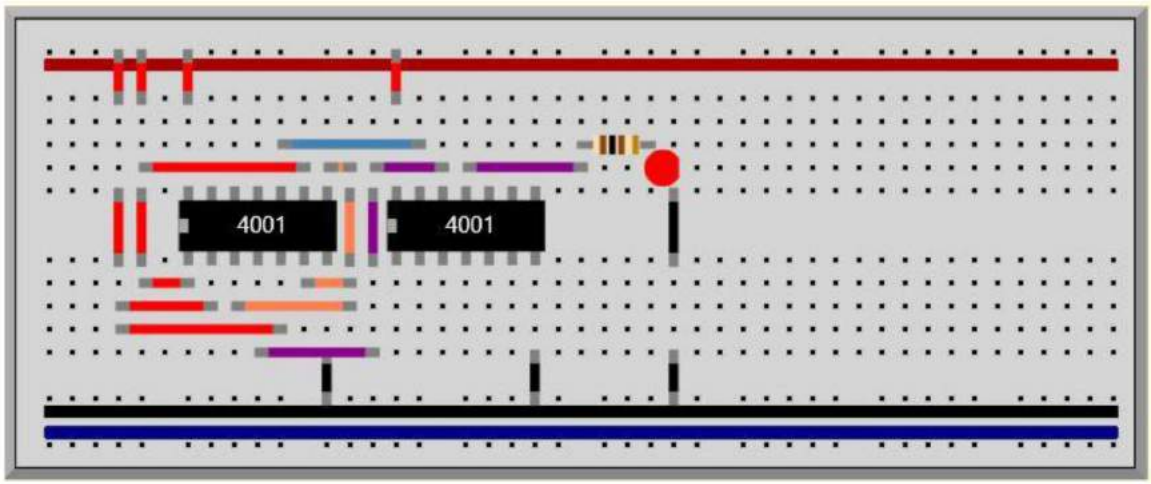


XNOR

Input:1 0 Output:1

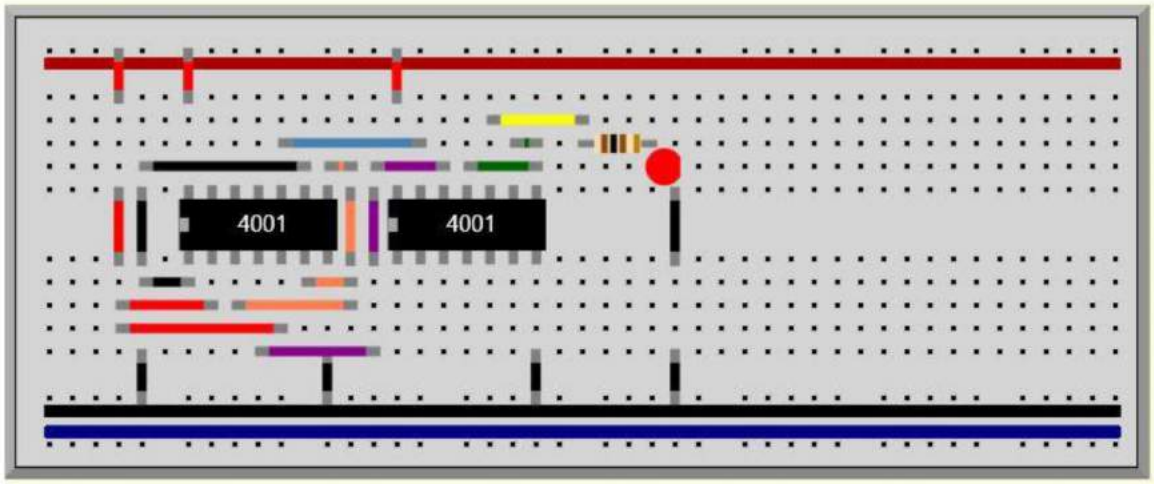


Input:1 1 Output:1



XOR

Input:1 0 Output:1



Input:1 1 Output:0